UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

"UMA PROPOSTA DE ALGORITMO ASSÍNCRONO PARA DIFUSÃO CONFIÁVEL ATÔMICA"

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA

LÍLIAM TEREZINHA RIBEIRO DUEÑAS

FLORIANÓPOLIS, DEZEMBRO, 1992

"UMA PROPOSTA DE ALGORITMO ASSÍNCRONO PARA DIFUSÃO CONFIÁVEL ATÔMICA"

LÍLIAM TEREZINHA RIBEIRO DUEÑAS

ESTA DISSERTAÇÃO FOI JULGADA PARA A OBTENÇÃO DO TÍTULO DE **MESTRE EM ENGENHARIA**
ESPECIALIDADE ENGENHARIA ELÉTRICA, AREA DE CONCENTRAÇÃO **SISTEMAS DE CONTROLE E AUTOMAÇÃO INDUSTRIAL**, E APROVADA EM SUA FORMA FINAL PELO CURSO DE PÓS-GRADUAÇÃO

Prof. Joni da Silva Fraga, Dr.
Orientador

Prof. Roberto Souza Salgado, PhD.
Coordenador do Curso de Pós-Graduação em Engenharia Elétrica

BANCA EXAMINADORA

Prof. Jean Marie Farines, Dr. Ing.

Prof. Joni da Silva Fraga, Dr.
Orientador

Prof. Raul Fernando Weber, Dr.

Prof. Vitório Bruno Mazzola, Dr.

Aos meus mui amados pais, Rafael e Terezinha, e meu marido, César Augusto, que em troca das minhas horas de ausência souberam me dar, com toda intensidade, o amor, a compreensão e o apoio, fundamentais para a execução deste trabalho.

AGRADECIMENTOS

À Deus, que se fez presente em todos momentos dificeis.

Ao professor Joni da Silva Fraga, pelo empenho e dedicação nos trabalhos de orientação e correção.

À todos os professores e amigos do Departamento de Engenharia Elétrica da UFSC, em especial, à engenheira Eliane Lúcia Bodanese, pela amizade e contribuição para a realização deste trabalho.

Aos funcionários da Vértice Sistemas Integrados, VSI, em especial ao engenheiro Ivan Carlos Ribeiro do Nascimento, pelo grande apoio e suporte computacional que muito contribuíram na edição e impressão deste trabalho.

À CAPES pelo suporte financeiro.

À todos que direta ou indiretamente contribuíram para que este trabalho fosse concluído, em especial aos meus irmãos Wiliam, Viviane e Flávio.

RESUMO

A necessidade de solucionar os problemas inerentes à comunicação em grupo, em sistemas distribuídos, motivou o desenvolvimento dos chamados "**Protocolos de difusão confiável**". Este trabalho retrata um estudo feito sobre estes protocolos e apresenta uma nova proposta.

O protocolo proposto atende às propriedades de acordo, ordenação total e terminação assíncrona mesmo na presença de faltas de omissão com grau M. Isto é, estações ausentes de menos de M engajamentos ("commitments") podem recuperar as mensagens mesmo na presença de R estações faltosas ("R-resilient system"), minimizando, em muitos casos, os custos de transferência de estados em modelos de processamento replicado na aplicação. Cada engajamento é feito em grupos de L mensagens, administrado por uma estação coordenadora (atual possuidora do token), com a participação da maioria das estações do grupo (configuradas num anel virtual). Isto diminui a necessidade de reconfiguração do anel, permite um maior paralelismo entre tratamento de faltas e execução do engajamento e minimiza os custos de controle ("overhead").

Sob condições normais de operação o protocolo requer duas fases para se completar e 3N mensagens-protocolo (N é o número de estações do grupo), de forma que para sistemas com valores de L altos o custo de controle tende a zero. A ordenação das mensagens é dada em função de suas ordenações nas emissoras e a localização das mesmas com relação à coordenadora. Uma análise de latência é apresentada mostrando que o tempo máximo de terminação é limitado e conhecido e da ordem de grandeza $O(N^2)$.

ABSTRACT

The need of solving the inherent problems of group communications in distributed systems has motivated the development of the so called **"Reliable Broadcast Protocols"**. This thesis presents a survey on these protocols and introduces a new proposal.

The proposed protocol satisfies the properties of agreement, total ordering and assinchronous termination, even in the presence of M-degree omission faults. That is, sites that are absent by less than M commitments can recover the messages even in the presence of R faulty sites (R-resilient system). This, in many cases, minimizes the state transfer costs in application-replicated processing models. Each commitment is done in groups of L messages, under the supervision of a coordinator site (the current token owner), with the participation of all other sites in the group (configured as a virtual ring). This reduces ring reconfigurations, it allows greater concurrency between fault processing and commitment executions and, minimizes overhead costs.

Under normal operating conditions, the new protocol needs two phases to complete and 3N protocol-messages (where N is the number of sites in the ring). Therefore, for systems with high values of L, control costs approaches zero. The message ordering is a function of the ordering at the emitting sites, and their location in the ring with respect to the coordinator. A latency analysis shows that it exists an upper bound to the termination time which is $O(N^2)$.

SUMÁRIO

LISTA DAS TABELAS

LISTA DAS FIGURAS

CAPÍTULO 1

INTRODUÇÃO

Sistemas informáticos distribuídos consistem de um conjunto de estações[1] que não compartilham memória, e só cooperam entre si através de trocas de mensagens pela rede de comunicação.

A forma mais elementar de sistemas distribuídos é a dita distribuição sequencial, onde um processo é particionado em módulos que são colocados em estações diferentes e executados sequencialmente. Isto é, a atividade "passea" por vários lugares do sistema. Este tipo de distribuição pode ser realizada através de interações bloqueantes, com protocolos de mensagens de pedidos e respostas de serviços, ou mais frequentemente, por chamadas a procedimentos remotos (RPC)[2] sem o menor problema.

Evoluindo na forma, tem-se programas distribuídos concorrentes estruturados no modelo cliente-servidor. Em tais modelos os processos clientes solicitam serviços dos processos servidores que os aceitam, respondendo a cada cliente individualmente.

Ainda em sistemas estruturados como cliente-servidor, encontra-se sistemas redundantes, para aplicações que requerem tolerância a faltas, e tempo de acesso aos dados reduzido. Tal redundância é introduzida pela replicação dos processos servidores, e as vezes dos processos clientes, em diferentes lugares do sistema. Neste caso tem-se vários processos mantendo, cada um, uma cópia de um conjunto de dados replicados. Tais réplicas de processos necessitam ter exatamente o mesmo conjunto de dados de entrada a todo instante, de forma que o processamento de uma conduza aos mesmos valores de estado e resultados dos processamento das outras réplicas no final de um certo periodo.

Diante destes sistemas redundantes, surgem algumas questões:

1. O que fazer quando um cliente precisa enviar uma mensagem para que **todos** os servidores executem um mesmo serviço? De início, parece trivial; principalmente com o uso de uma rede confiável de comunicação ponto-a-ponto. Bastaria que o cliente enviasse a mensagem contendo os dados para cada um dos servidores. Mas, se o emissor falhasse tendo enviado a mensagem a somente alguns dos servidores ter-se-ia um conjunto de processos com estados inconsistentes. A solução para este tipo de problema é que um ou mais dos servidores que

---

[1] Cada estação podendo conter um ou mais processos autônomos.

[2] O conceito de RPC, ou "Remote Procedure Call", foi introduzido por Birrell e Nelson (1984). A RPC é uma primitiva de comunicação equivalente a uma chamada de procedimento, e como tal, só pode ser usada entre UM processo chamador e UM processo chamado.

receberam a mensagem detectem a falha do emissor e a retransmitam aos servidores restantes.

2. Para resolver a questão acima é preciso que os servidores mantenham uma cópia da mensagem até terem certeza que todos a receberam. Fica claro que, por uma questão de custo, as mensagens não podem ser armazenadas para sempre. Logo, é necessário se descobrir o momento em que as mensagens podem ser removidas. Isto gera um novo problema: uma segunda cópia da mensagem pode chegar após todo conhecimento da primeira ter sido removido e por isto ser adicionada, erroneamente, uma segunda vez pelo servidor. Portanto, para uma mensagem ser removida, deve-se garantir que ela foi retirada de todos os servidores.

3. Sejam agora dois clientes A e B enviando, cada um, uma mensagem aos servidores. Se por algum motivo (como por exemplo, corrupção da mensagem por interferência electromagnética ou falta de espaço para armazenamento em alguns servidores) a mensagem de A não for recebida por alguns servidores, que só detectem a falta e a adquiram após a recepção da mensagem de B, um novo problema é estabelecido: os servidores que foram replicados para a tolerância a faltas, apresentam agora inconsistência em seus estados. Para resolver este problema pode-se criar mecanismos que garantam que as mensagens serão entregues a todos os servidores numa mesma ordem.

Um processamento em múltiplas replicas é apenas um exemplo que evidencia a importância e a não trivialidade dos problemas de comunicações em grupo em sistemas distribuídos. Os problemas citados poderiam ser extendidos para: alocação de recursos, migração de processos, consenso distribuído, "scheduling" e balanceamento de cargas. As soluções para estes problemas estão fundamentadas no que tradicionalmente é chamado na literatura de "**protocolos de difusão confiável**".

O objetivo inicial deste trabalho foi desenvolver e implementar um serviço de difusão confiável no sistema ADES [Fraga 89], criando desta forma um suporte para o desenvolvimento de modelos de tolerância a faltas fundamentado em processos replicados. Este trabalho tomaria como base as principais propostas de protocolos de difusão confiável presentes na literatura. Neste estudo, os algoritmos seriam examinados no sentido de determinar aqueles que melhor se adequassem às condições de aplicações em tempo real. Esta adequação, senão num sentido mais estrito da taxonomia de tempo real, deveria se dar num sentido de baixa sobrecarga ("overhead") de mensagens de controle e, sobretudo de um algoritmo limitado no tempo.

Vários aspectos destes algoritmos foram retomados no sentido de adaptá-los às condições do ambiente ADES. Na evolução deste trabalho, no entanto, notou-se que a proposta inicial foi determinando contornos próprios ao serviço desejado, instigando assim, a se apresentar uma proposta de algoritmo de difusão confiável. Este algoritmo incorpora o que se crê haver de mais adequado na bibliografia, para a classe de aplicações a que se destina, além de outras soluções reputadas como originais.

A constatação se o algoritmo atende aos requisitos colocados acima deve ocorrer com base nos próximos capítulos e nos trabalhos de simulação realizados. Para tal, esta dissertação se apresenta da seguinte forma :

O capítulo 2 introduz os principais conceitos existentes de protocolos de difusão confiável e alguns dos mais relevantes protocolos de difusão confiável atômica da literatura.

O capítulo 3 introduz um modelo geral de protocolo de difusão confiável e apresenta o protocolo proposto pelo autor.

O capítulo 4 apresenta uma discussão sobre os parâmetros utilizados no algoritmo e suas relações com os custos deste.

O capítulo 5 apresenta a especificação desta proposta de protocolo na linguagem Estelle* e os resultados referentes à validação desta, a partir da simulação da especificação Estelle*, através da ferramenta ESTIM.

O capítulo 6 apresenta, por fim, as conclusões mais importantes e as perspectivas que se abriram com este trabalho.

CAPÍTULO 2

PROTOCOLOS DE DIFUSÃO CONFIÁVEL

2.1 - Introdução

Este capítulo explora os principais aspectos envolvidos com a comunicação em grupo. Neste sentido, nos ítens subsequentes é apresentada uma conceituação referente a protocolos de difusão confiável e comunicação em grupo. Uma revisão da literatura é apresentada a seguir, onde as características dos principais protocolos são evidenciadas a partir de aspectos como desempenho (tempos máximos envolvidos), sobrecarga ("overhead": o número de mensagens-protocolo por mensagem difundida), coordenação, ordenação, etc. Antes, será apresentada uma classificação de faltas segundo seus efeitos, de modo que se possa discutir os algoritmos segundo suas propriedades, envolvendo a tolerância a faltas.

2.2 - Aspectos gerais em comunicação por difusão

2.2.1 - Classificação de Faltas

Em sistemas distribuídos a única conduta visível de um processo é a sequência de mensagens que este envia. Um processo é dito operacional se sua conduta se faz conforme seu algoritmo. Uma falha ocorre se um processo desvia de seu algoritmo fornecendo um serviço inapropriado, isto é, propagando um erro causado por uma falta no processo. Processos propagando erros são ditos "**processos faltosos**".

A literatura é rica em descrições de modelos de tolerância a faltas. Estes modelos formam uma hierarquia de severidade crescente quanto aos tipos de faltas toleradas. Em geral, algoritmos tolerantes a faltas tornam-se mais complicados e caros para executar quanto menos restritivas são as faltas que eles toleram. Assim, os tipos de faltas assumidas quando um algoritmo é projetado são determinantes para o custo do projeto e o desempenho do mesmo. A classificação apresentada neste item considera os efeitos das faltas e está de acordo com classificações apresentadas em [Powell 90], [Veríssimo 89], e [Nacamura 92].

Faltas no domínio dos valores

As **"faltas de valores"** determinam que um serviço entregue no instante esperado apresente seu valor fora da faixa de valores para ele aceitáveis. Os erros produzidos por esta classe de faltas são, por exemplo, envio ou entrega de mensagens com conteúdo modificado.

Faltas no domínio do tempo

As **"faltas no domínio do tempo"** são basicamente faltas de temporização. As faltas de temporização determinam que os valores corretos de um serviço sejam recebidos fora do intervalo especificado para o mesmo. Tais faltas podem produzir:

- **Erros por atraso**: onde a recepção do serviço ocorre após um tempo máximo de espera e,
- **Erros por antecipação**: onde a recepção ocorre antes de um tempo mínimo de espera.

Um caso particular de faltas de temporização é aquele em que uma ativação de serviço nunca tem seu valor liberado. Este tipo de falta, denominado **"falta por omissão"**, pode ser interpretado como produzindo um erro por atraso onde a recepção desta ativação de serviço se dará em instante de tempo tendendo a infinito. Se após a primeira omissão o serviço deixa de responder sistematicamente a futuras ativações, o serviço é dito incorporando a semântica de **"faltas de crash"**.

Se o serviço apresenta uma semântica de omissão incorporada, com **"grau de omissão K"**, no seu comportamento é permitido a omissão de algumas respostas (K omissões). Porém, ao atingir o limite de K omissões, todas as suas respostas subsequentes deverão ser omitidas, caracterizando então o "crash" deste serviço.

Faltas arbitrárias

As **"faltas arbitrárias"** envolvem erros nos domínios de valores e de tempo. Algumas classes de faltas arbitrárias têm merecido atenção especial na literatura [Lamport 82], [Powell 90], [Veríssimo 90], [Cristian 85] : [Nacamura 92]

- **Faltas maliciosas ou intencionais**: estas faltas geram serviços fora do domínio de valores em instantes aleatórios. Estes dados mantêm ainda alguma propriedade de modo a confundi-lo com valores válidos no instante considerado. O comportamento de componentes com semântica de faltas maliciosas corresponde ao que se pode ter de mais indesejável em um sistema.

- **Faltas de improvisação**: um erro de improvisação ocorre quando o sistema entrega espontaneamente o serviço em momento não esperado.

A figura 2.1 sintetiza a classificação de faltas apresentada, indicando as relações entre os tipos citados.

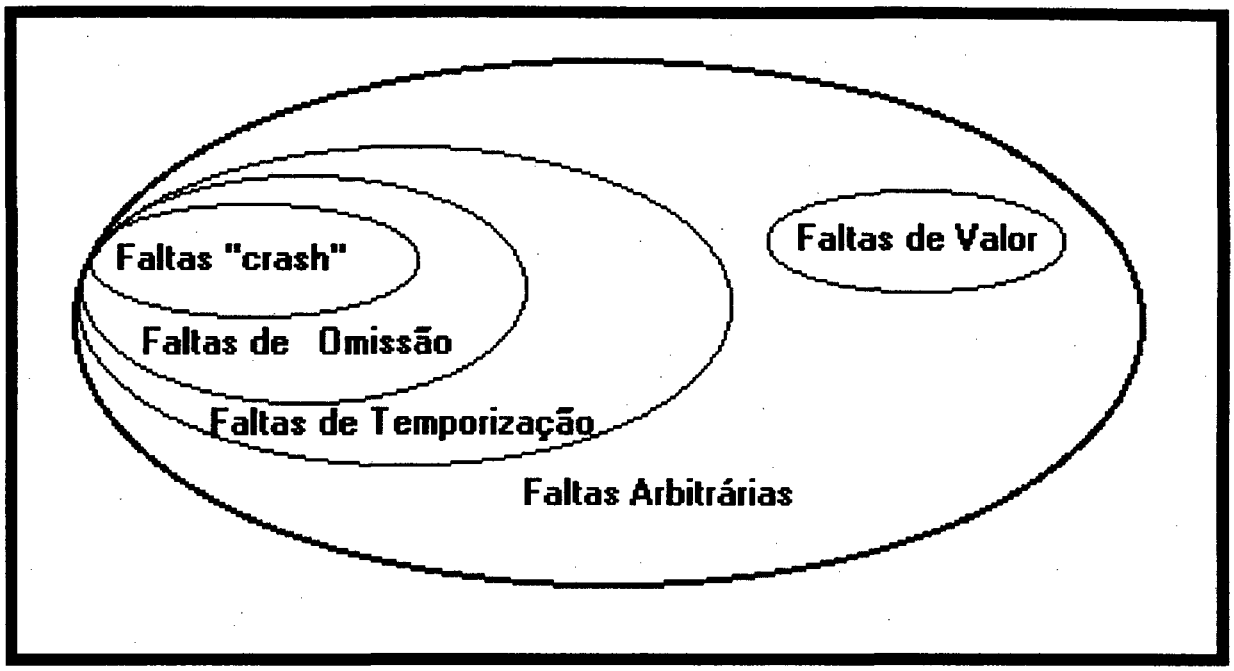


Figura 2.1 - Classes de faltas

2.2.2 - Protocolos de Difusão Confiável: conceituação

Um protocolo de difusão confiável ou "Reliable Broadcast" é um serviço responsável por garantir, mesmo na presença de faltas, um comportamento bem definido em relação a mensagens difundidas no sistema. Subentende-se aqui como difusão de uma mensagem o envio desta a um grupo de processos ("broadcast" ou "multicast", dependendo se o grupo contém todos ou parte dos processos).

Em um sistema distribuído pode-se ter vários grupos, sobrepostos ou não, segundo as funcionalidades definidas no sistema, suportados então, pelo serviço de comunicação de grupo.

As mensagens **recebidas** em um participante do grupo só são consideradas **aceitas**, e passadas ao usuário do serviço, quando satisfazem todas as condições do protocolo de difusão. Este ato de "aceitação" é chamado de **engajamento** da mensagem (**message commitment**). O engajamento da mensagem dependerá, portanto, das propriedades envolvidas no protocolo utilizado.

A condição de protocolo de difusão confiável está fundamentada em três propriedades básicas [Cristian 85]: acordo, ordenação e terminação. Conforme o maior ou menor rigor destas propriedades tem-se classes diferentes de protocolo de difusão confiável.

Acordo

A propriedade de acordo (ou atomicidade) deve garantir, mesmo em presença de faltas, ou que todos os participantes corretos recebam uma mensagem difundida ou que nenhum destes a aceite. Porém, o espectro de faltas, como visto no item anterior, é bem amplo, envolvendo faltas de "crash", omissão, temporização e ainda, faltas arbitrárias (incluindo as de valor). Assim os efeitos e a dificuldade na detecção dos tipos de erros provocados por estas faltas conduzem a uma hierarquia dos tipos de faltas em ordem crescente da severidade:

Faltas "crash" $\subset$ Faltas de Omissão $\subset$ Faltas de Temporização $\subset$ Faltas Arbitrárias

O que implica que, quanto mais restritivas forem as hipóteses de faltas, menos complexo e de menor custo é a concepção e a implementação de algoritmos dos serviços considerados. Porém serão menos confiáveis estas implementações.

As proposições de algoritmos de comunicação de grupo (algoritmos distribuídos) apresentados na literatura, atuam de forma diferenciada neste espectro. Assim, enquanto os protocolos de [Birman 87], por exemplo, suportam apenas faltas "crash", os protocolos propostos em [Perry 86], [Chang 84] e [Luan 90] garantem a propriedade de atomicidade na comunicação, na presença de faltas de omissão (que incluem o "crash"). Já a proposição de [Lamport 85], é um exemplo clássico de protocolos de comunicação que garantem a entrega de mensagens na presença de faltas arbitrárias (atendem a todo espectro de faltas). Protocolos que atendem, como este último, hipóteses mais gerais de faltas devem se utilizar de técnicas de acordos normalmente mais complexas: " Byzantine agreement " [Lamport 82].

Em outro sentido, alguns protocolos têm sido propostos com o objetivo de relaxar a propriedade de atomicidade. Para isto foram definidas algumas semânticas de comunicação menos restritivas [Delta4 90]:

- **"at least K"** qualquer mensagem entregue a um participante deve ser entregue a no mínimo K participantes corretos;
- **"best effort K"** na ausência de faltas, qualquer mensagem entregue a um participante, deve ser entregue a K participantes.

Estas duas semânticas enfraquecem as necessidades para o engajamento da mensagem. Já não é necessário todos os participantes terem a mensagem em um dado tempo mas um número mínimo de participantes (k).

Ordenação

A propriedade de "ordenação" garante que todas as mensagens emitidas em um grupo serão recebidas por seus participantes numa mesma ordem. Neste sentido, a ordenação vem complementar a propriedade de acordo, dando uma visão única aos participantes deste grupo sobre o conjunto de mensagens difundidas.

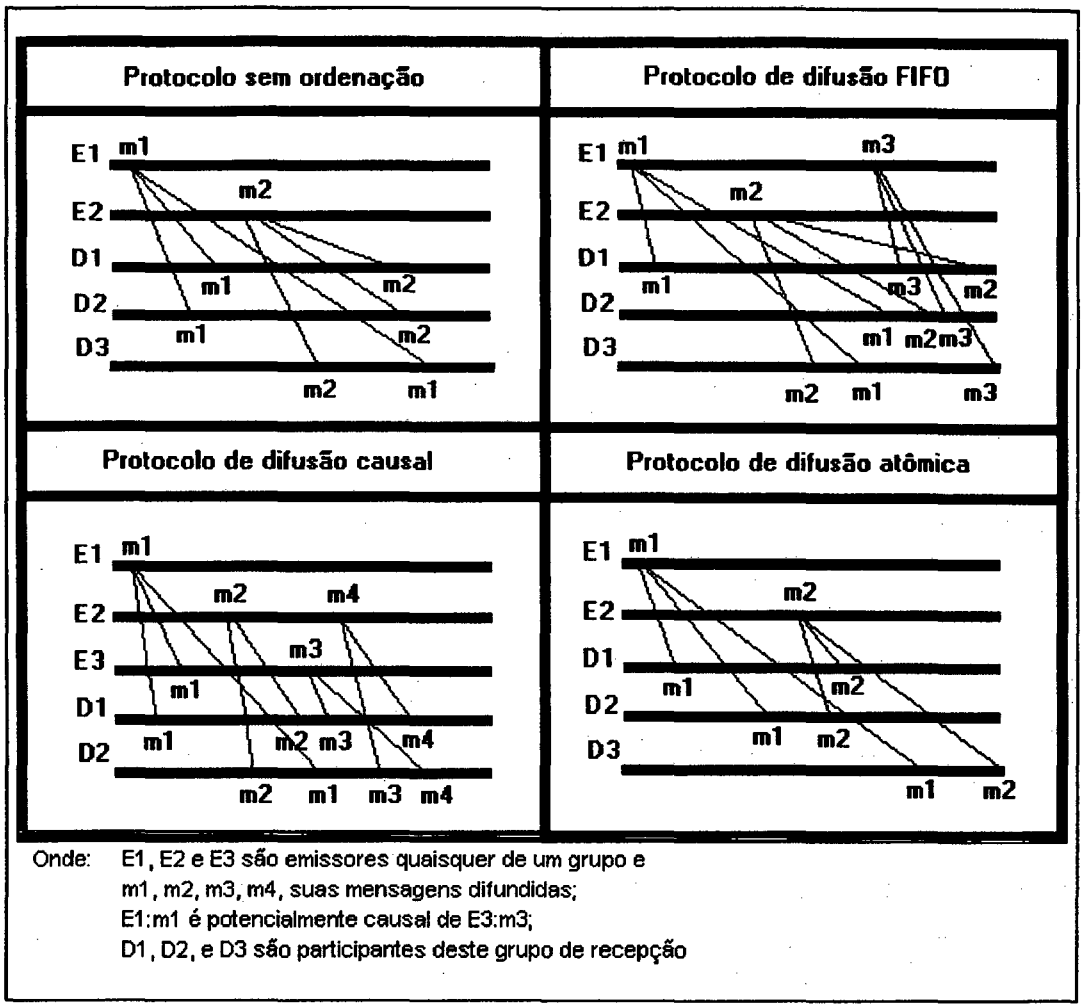


Figura 2.2 - Classificação dos protocolos segundo a sua ordenação

Alguns dos protocolos, como o de [Chang 84] e o ABCAST de [Birman 87] fornecem a ordenação total de mensagens outros, como o de [Babaoglu 88], apresentam uma ordenação relaxada a tipo FIFO. Assim, os protocolos de difusão podem ser divididos em classes, segundo o tipo de ordenação usado [Shivastava 91] :

- **Protocolos sem ordenação**: mensagens difundidas são engajadas sem nenhum critério de ordenação.

- **Protocolo de difusão FIFO**: nesta classe as mensagens difundidas por uma mesma estação serão engajadas na mesma ordem de emissão. Mensagens difundidas por estações distintas podem ser recebidas por diferentes participantes em ordens opostas.

- **Protocolos com ordenação causal**: protocolos desta classe extendem a ordenação do tipo anterior para causalidade potencial no envio de diferentes emissores. Assim, em situações onde dois emissores E_1 e E_3 emitem respectivamente as mensagens m_1 e m_3. Se a emissão m_1 provocar potencialmente a emissão m_3, a "ordenação causal" deve garantir que todos os participantes receberão m_1 antes de m_3, caso contrário, tais mensagens não serão recebidas de forma idêntica por participantes distintos. Os protocolos que atendem esta ordenação são chamados de protocolos de difusão causal.

- **Protocolos com ordenação total**: neste caso, todas as mensagens difundidas são engajadas numa mesma ordem por todos os participantes. Esta ordem de engajamento não é necessariamente a ordem cronológica de emissão das mensagens. Alguns autores denominam esta classe de protocolos de "**protocolo de difusão atômica**".

A figura 2.2 sintetiza os protocolos classificados segundo estes aspectos de ordenação.

Terminação

A propriedade de terminação esta ligada a que cada participante saiba o resultado do engajamento de uma mensagem dentro de um tempo limitado.

Em [Veríssimo 89] são definidos alguns parâmetros que servem para caracterizar a terminação de um protocolo:

- **Tempo de Execução (T_e)**: é o intervalo de tempo entre o pedido de enviar uma mensagem m_i e a confirmação da última indicação de recebimento da mesma pelos participantes do grupo.

- **Tempo de Inconsistência (T_i)**: é o maior intervalo de tempo entre indicações de recebimento da mensagem pelos participantes destinatários.

Por definição, T_i mede o tempo entre a primeira entrega e a entrega a todos os participantes. Por definição $T_i \leq T_e$ já que T_e corresponde ao tempo envolvido desde a emissão até o recebimento por todos os participantes da mensagem. Os tempos definidos são usados em [Veríssimo 89] para definir medidas de sincronismo de protocolos de difusão confiável :

- **Rigidez**: " Dado um protocolo de difusão confiável, este é dito "Rígido" se existir τ, tal que para cada execução:

$$|T_i| \leq \tau, \quad \text{com } \tau > 0 \text{ "}$$

- **Estabilidade**: " Dado um protocolo de difusão confiável, este é dito "Estável" se existir σ, tal que, para cada duas execuções quaisquer:

$$|T_{e1} - T_{e2}| \leq \sigma, \quad \text{com } \sigma > 0 \text{ "}$$

Com estas duas medidas fica fácil estabelecer critérios para se classificar protocolos segundo seus graus de sincronismo. Um protocolo é tanto mais síncrono quanto menor forem τ e σ quando comparados a T_e. Assim protocolos completamente rígidos ($\tau = 0$) e estáveis ($\sigma = 0$), são ditos **Totalmente Síncronos**.

Os protocolos que se aproximam da rigidez e estabilidade completa, isto é, τ e σ desprezíveis em relação a T_e, são ditos com **Sincronismo Forte**. É o caso, por exemplo de [Cristian 85]. Relaxando os valores de rigidez e estabilidade para valores não tão pequenos em relação a T_e (embora limitados e conhecidos), tem-se os protocolos ditos de **Sincronismo Fraco** (exemplo: [Peterson 88]).

No sentido oposto, apresentam-se os protocolos **Assíncronos**, que têm a rigidez e a estabilidade não limitados. Destes protocolos é requerido apenas que terminem. Nesta situação, é assumido que, existindo t_e e r [Veríssimo 90],

$$Pr \{ T_e > t_e \} < r , \quad \text{para } r > 0.$$

Esta condição garante apenas que o protocolo eventualmente termine. Os protocolos assíncronos, atendendo apenas a esta condição, não são adequados para aplicações em tempo real. Diante disto, é necessário assumir uma condição de T_e limitado. Considerando então $t_e=T_e(max)$, tem-se:

$$Pr \{ T_e \rangle T_e(max) \} \simeq 0.$$

Esta condição, então, garante o tempo de execução limitado, mas não o suficiente para garantir a adequação destes protocolos para aplicações em tempo real. Valores muito grandes de T_e (latência de mensagem) implicarão no baixo desempenho do protocolo. Exemplos de protocolos assíncronos estão em [Chang 84],[Birman 87] e [Luan 90].

2.3 - Alguns protocolos existentes

Atualmente existe um vasto número na literatura de proposições de protocolos de difusão. A seguir são apresentados alguns destes protocolos considerados como os mais representativos.

2.3.1 - Protocolo de Difusão Confiável de Chang

O protocolo de difusão de [Chang 84] apresenta características de protocolo assíncrono com ordenação total, suportando propriedades de acordo, de ordenação e de terminação, em presença de faltas por omissão. O "commit" de mensagem é centralizado sobre o conceito de token circulante. A estação possuidora do token é denominada "**token site**" e é a responsável, enquanto possuidora do token, pelo reconhecimento das mensagens difundidas, pela retransmissão de mensagens quando requisitado por algum receptor e também, pela transferência do token para o próximo token site. As estações operacionais do sistema são relacionadas em uma lista denominada "**token list**" que determina o anel virtual para a transferência do token. O protocolo de Chang funciona em duas fases: Normal e Reconstituição (Reformation).

Fase normal

Uma estação ao difundir uma mensagem B(s,n), onde s é o endereço do emissor e n o número da mensagem emitida por s, fica aguardando o reconhecimento do token site. O token site, a cada nova

mensagem recebida, após verificar as informações de controle, difunde o reconhecimento. O "acknowledgement" da mensagem n de s resulta na difusão de um número associado a esta mensagem: o "**timestamp**". Os receptores, de posse destas mensagens de reconhecimento, têm uma informação global, o timestamp, que é o número de sequência da mensagem n no sistema. Diante destes mecanismos os receptores têm plenas condições de estabelecer ordens sobre a fila de mensagens recebidas, tendo como base os timestamps atribuídos pelo token site. A descontinuidade nas sequências possibilita às estações receptoras detectarem o não recebimento de mensagem (mensagens perdidas).

As mensagens são repostas pelo token site sob demanda das estações receptoras que detectaram descontinuidade nas suas ordens de recepção. As mensagens terão copias no token site, até que seja concluído o acordo sobre o seu engajamento.

Afim de evitar que o token site falhe perdendo mensagens que tenham sido reconhecidas, mas ainda não engajadas é feita a rotação do token pelas estações do anel presentes no "token list". O token é transferido como parte do acknowlegdment de uma mensagem difundida e só é aceito pelo novo token site após este recuperar todo o contexto de mensagens reconhecidas em processo de engajamento. O processo de engajamento de uma mensagem está baseado na circulação do token por L+1 estações contidas no token list. Isto garante que, mesmo que L estações venham a falhar, todas as mensagens engajadas serão recuperadas na fase de reconstituição (Sistema "L-resistente").

Fase de Reconstituição

O protocolo entra nesta fase quando uma falha é detectada ou uma estação é recuperada. Nesta fase uma nova token list é gerada, mas é garantido que nenhuma mensagem já engajada da antiga token list será perdida.

2.3.2 - Protocolo de Difusão Confiável de Birman

O protocolo proposto em [Birman 87] é da classe de protocolo com ordenação total e comunicação de grupo assíncrono, mesmo em presença de faltas de "crash". O protocolo de difusão ordena as mensagens endereçadas aos usuários colocado-as no "delivery queue" de cada processo usuário. O engajamento se dá pela retirada das mensagens do "delivery queue" em ordem FIFO.

Os tipos de ordenação utilizados na "delivery queue" na verdade definem dois protocolos de difusão diferentes: ABCAST ("Atomic Broadcast") e CBCAST ("Causal Broadcast").

ABCAST

Este protocolo é usado em aplicações que exigem ordenação total. O ABCAST opera baseado em "timestamps" que são associados a cada mensagem no sistema. A ordem total deverá ser estabelecida tendo como base os valores de timestamp.

Cada mensagem que chega num receptor participante, é antes armazenada em uma fila de mensagens pendentes (fila provisória) e etiquetada como "indisponível". O emissor desta mensagem deve receber propostas de timestamp dos receptores participantes, enviados como reconhecimento da mensagem. O emissor, após coletar as propostas de timestamp, deve calcular o de maior valor e assumí-lo como o "timestamp final" da mensagem.

De posse do "timestamp final" os receptores mudam a etiqueta da mensagem para "disponível" e colocam-na na ordem segundo o valor do timestamp na fila de mensagens disponíveis. Esta fila de mensagens disponíveis são passadas à "delivery queue" na ordem de seus timestamps.

CBCAST

Este protocolo é usado em aplicações onde somente a ordenação Causal deve ser garantida. Isto é, CBCAST é usado para forçar uma ordem entre duas difusões A e B, quando uma das mensagens (A) é potencialmente causal da outra (B) e esta relação de potencialidade causal é significante ($A \xrightarrow{c} B$).

No CBCAST para cada processo P, há um buffer BUF_p o qual contém cópia das mensagens enviadas de um emissor E para P, bem como cópia de mensagens que chegam a P na rota para outros processos. Para uma mensagem B ser transmitida do BUF_p na estação S para o BUF_q na estação T é criado primeiro um pacote de transferência $<B_1, B_2, ...>$ que inclui todas as mensagens. Este pacote deve garantir a ordem causal; considerando valores de timestamp para forçar esta ordenação; então seja o pacote $<B_1, B_2, ..., B_i, ...>$, onde B_i é uma mensagem com timestamp i. Se $B_i \xrightarrow{c} B_j$ (B_i antecede B_j na ordem causal), então $i < j$. O pacote transferido deve forçar esta ordenação.

Na recepção de um pacote, para cada i são executados os seguintes passos:

- Se B_i é duplicata, a mensagem é descartada;
- Se o processo q pertence aos destinatários da mensagem B_i, então B_i é colocada no "delivery queue". A cópia de B_i que permanece em BUF_q (e que posteriormente vai ser enviada para outra estação), deve ser atualizada retirando q de sua lista de destinatários.
- Qualquer mensagem produzida por q, após a recepção de B_i na "delivery queue"; deve receber um timestamp maior que i e os pacotes formados posteriormente deverão levar em consideração estes valores de timestamp (ordem causal).

Desta forma a mensagem é difundida garantindo a potencialidade causal com a mínima necessidade de sincronização no algoritmo de difusão.

GBCAST

O protocolo de [Birman 87] apresenta ainda funções para informar aos membros operacionais do grupo de processos quando um membro falhou, reintegrou-se, aderiu ou saiu voluntariamente, ou ainda, quando houve alguma mudança de propriedades globais. Cada membro do grupo mantém informações

sobre este grupo em uma "view" . As trocas envolvendo o protocolo GBCAST é implementado no sentido de atender estas funções de gerenciamento de grupo.

2.3.3 - Protocolo de Difusão Confiável de Cristian

O protocolo apresentado em [Cristian 85] na verdade representa uma familia de protocolos de difusão atômica que resistem a erros de severidade progressiva . Estes protocolos se baseiam em difusão sobre uma rede ponto a ponto de redundância espacial. As propriedades de acordo, de ordenação e de terminação são suportadas através de uma execução fortemente síncrona dada pela sincronização dos relógios locais. Os protocolos síncronos (e/ou fortemente síncronos) ao invés de trocas de mensagens-protocolos, usam a passagem de tempo e o conhecimento dos limites temporais para obter as informações para o acordo e a ordenação (baseados no comportamento de pior caso). Assim, todas mensagens são engajadas "simultaneamente" na mesma ordem por todos receptores, com base no tempo t em que as mensagens foram enviadas (indicado por um timestamp) e no atraso máximo (Δ) que uma mensagem pode levar para chegar em todos seus destinos.

A técnica de difusão de mensagem usada no protocolo consiste de:

1. Uma estação operacional envia a mensagem a todos seus links (difusão)

2. Quando uma nova mensagem correta é recebida em algum link de uma estação operacional, deve ser armazenada num buffer de recepção (recepção) e retransmitida aos demais links (redifusão)

3. Após um "tempo de terminação" ($t + \Delta$) é garantido que a mensagem chegou a todos os seus destinos devendo ser engajada (engajamento)

Para realizar a propriedade de ordenação é suficiente que todos processos corretos engajem na ordem gerada pelos timestamps e que mensagens geradas em tempos de clocks iguais sejam engajadas em ordem crescente dos identificadores dos emissores.

Para difundir uma mensagem a estação emissora assinala antes um timestamp indicando sua identificação como estação e o instante de emissão t. Como nos protocolos síncronos os relógios de estações corretas, a menos de ε, são exatos, uma mensagem enviada por uma estação P, no instante t_p, para uma outra Q, via (h-1) estações intermediárias (sobre h links corretos) deve chegar em q no tempo t_q, tal que:

$$\varepsilon \leq t_q - t_p \leq \varepsilon + h\delta$$

onde δ é o tempo de difusão entre duas estações vizinhas.

Afim de determinar o atraso máximo (Δ) que uma mensagem pode demorar para chegar em todos os seus destinos (Tempo de Terminação) deve-se levar em conta os tipos de faltas suportados em cada protocolo.

Protocolo Tolerante a Faltas por Omissão

Este protocolo, afim de garantir a propriedade de acordo, necessita apenas que uma estação, ao receber uma mensagem difundida em t, espere por um atraso máximo (Δ) antes de engajá-la. Assim, as estações receptoras, ao receberem a mensagem, verificam se o tempo decorrido entre a emissão e a recepção é inferior a Δ e se a mensagem ainda não foi recebida nenhuma vez, caso afirmativo a mensagem é armazenada num buffer de recepção e (re)difundida para todas as demais estações. Ao fim de $t + \Delta$ todas as mensagens difundidas no tempo t são engajadas pela ordem de prioridade de suas emissoras.

O atraso máximo (Δ) é dado como:

$$\Delta = \pi\delta + D_t + \varepsilon$$

onde π é o número máximo de faltas suportadas no sistema (grau de resistência), de forma que o termo $\pi\delta$ corresponde ao pior caso de atraso entre a emissão da mensagem e a recepção desta pela primeira estação correta. O termo D_t, indica o tempo para esta estação correta difundir a mensagem às demais estações e o último termo (ε) corresponde à incerteza de sincronização entre os relógios locais.

Protocolo Tolerante a Faltas de Temporização

Para garantir a propriedade de acordo na presença de faltas de temporização, é estipulado o intervalo de tempo em que as mensagens podem ser aceitas. Ou seja, se no tempo τ a estação recebe uma mensagem com timestamp t que foi retransmitida sob h links intermediários, esta mensagem só poderá ser aceita se:

$$t - h\varepsilon \leq \tau \leq t + h(\delta + \varepsilon)$$

Assim, afim de controlar o número de retransmissões já ocorridas, a mensagem possue em suas informações de controle um contador K que é incrementado por cada estação que faz sua (re)transmissão.

Desta maneira, uma mensagem pode levar até $\pi(\delta + \varepsilon)$ unidades de tempo na rede antes de ser aceita pela primeira estação operacional. Por isto o atraso máximo é dado por:

$$\Delta = \pi(\delta + \varepsilon) + D_t + \varepsilon$$

Protocolo Tolerante a Faltas Maliciosas

As faltas maliciosas são toleradas, neste protocolo, através da autenticação das mensagens trocadas pelas estações durante uma difusão. Assim, as mensagens ao serem difundidas são autenticadas, de maneira única e inequívoca, por suas emissoras e co-autenticadas por cada estação retransmissora.

Após recebida a mensagem, esta é checada quanto à sua autenticação. Se a mensagem não foi corrompida é verificada a lista de co-autenticantes para garantir que nenhuma assinatura foi duplicada (em caso de duplicação a mensagem é descartada). Não havendo duplicação de autenticação ou falta de

temporização (h é dado pelo número de co-autenticantes) é verificada a existência da mensagem no buffer de recepção. Se a mensagem já existe no buffer, e possuem um único valor em ambas as cópias, esta é descartada. Se existe, mas com valor diferente, a emissora é considerada faltosa. Neste caso a mensagem é redifundida (após co-autenticada) para que as demais estações percebam a falta. Se a mensagem é uma nova difusão a estação a coloca no buffer de recepção e a retransmite aos demais links após co-autentica-la.

Como o tempo de autenticação é desprezível face aos demais, o atraso máximo (Δ) é o mesmo dado para o protocolo tolerante a faltas de temporização.

2.3.4 - Protocolo de Difusão Confiável de Luan/Gligor

O protocolo de difusão apresentado em [Luan 90] é um protocolo assíncrono com ordenação total que suporta faltas por Omissão. As mensagens são recebidas por todas estações numa única ordem determinada por decisão de consenso de maioria em cada rodada de engajamento. Em cada execução de protocolo pode ocorrer duas condições : Normal e de Terminação. Em condição Normal, o protocolo faz o engajamento de uma lista de mensagens após o consenso e tolera um número significante de estações e links falhos. A Terminação é invocada por qualquer estação que não possa terminar a execução dentro de um dado período de tempo devido à falha da estação iniciadora do consenso ou do link entre a estação iniciadora do consenso e esta.

Uma estação pode iniciar o consenso sempre que seu buffer de mensagens estiver cheio ou quando um dado período de tempo decorrer desde a última vez que esta estação/processo engajou uma lista de mensagens. A estação que inicia a execução é denominada "**iniciadora**".

A camada de serviço de difusão de mensagens é implementado a partir do "Processo tratador de mensagens" (MSP) e de memórias para armazenamento:

- "Memória para emissão de mensagens" MSQ - memória não volátil;
- "Memória para a recepção de mensagens aceitas" (MRQ) - memória não volátil;
- "Buffer de mensagens recebidas" (MRB) - memória volátil.

Assim mensagens da aplicação são enviadas para MSQ e retiradas de MRQ. Mensagens difundidas pela rede são recebidas pelo protocolo em MRB e, após uma execução do protocolo, engajadas e colocadas em MRQ. Um MSP_i contém as seguintes informações:

- **BCL_i (" Buffer Content List")** - Lista das mensagens contidas em MRB do MSP_i.
- **$COML_i$ ("Commit List")** - Lista com a sequência de mensagens que já foram engajadas por MSP_i.

Condição Normal

O protocolo sob condição Normal consiste de três fases: Convite, Notificação e Engajamento.

Convite: quando o convite é recebido pelos outros MSP's e, se estes ainda não tiverem aderido à execução de outro iniciador, o COMLi do iniciador é comparado com o dos MSP's receptores. Neste caso duas situações podem ocorrer:

- O $COML_i$ do receptor é maior do que o do iniciador: neste caso o iniciador perde o consenso e não estará apto a iniciar uma execução de protocolo enquanto não engajar as mensagens que faltam na sua lista $COML_i$.
- O MSP do receptor adere à execução, respondendo positivamente ao iniciador: a mensagem de reconhecimento deve levar o BCL do receptor ao iniciador.

Notificação: o iniciador coleta então as respostas dos outros MSP's. Duas situações podem então ocorrer:

1. Uma **resposta negativa** pode ser recebida: o iniciador deve então abortar a execução e enviar um "catch-up" aos MSP's que responderam positivamente ao convite. Os MSP's que receberam o "catch-up", juntamente com o iniciador, devem adquirir e/ou engajar as mensagens que aparecem no COML do MSP negador que ainda não foram engajadas pelos mesmos.
2. Recepção de **respostas positivas** de todos MSP's ou **esgotamento** de um **timeout** sem respostas negativas: neste caso se [N/2 + k] MSP's responderam positivamente, o iniciador calcula duas listas[1]:
   - **MCSL ("Maximal Common Subset List")**: é uma lista calculada tomando o máximo subconjunto comum de [N/2 + K + 1] BCL's dos MSP's convidados. A ordem das mensagens desta lista é indicada arbitrariamente pelo iniciador.
   - **$COHL_{init}$ ("Initial Cohort List")**: é uma lista dos [N/2 + K + 1] MSP's que formaram a MCSL (denominados **escravos**). A $COHL_{init}$ é usada no protocolo de terminação quando o iniciador cai ou é inacessível.

   Tais listas MCSL e $COHL_{init}$ são difundidas para todos os MSP's e os MSP's pertencentes a $COHL_{init}$ respondem com um ACK.

[1] N é o número de MSP's da rede e K é o valor da margem de segurança que permite que mesmo se K estações ou links de comunicação falharem durante as fases de notificação e de engajamento, as mensagens ainda poderão ser engajadas (sistema K-resistente).

Duas exceções são possíveis na rodada de convite: menos de [N/2 + K] MSP's escravos respondem ao convite. Nesta caso, o iniciador difunde um "abort" para todos MSP's e termina a execução. Se os MSP's não recebem nenhuma resposta do iniciador dentro de um timeout, estes abortam a execução.

Engajamento: Se mais de [N/2 + 1] ACK's são recebidas, é formado o que o protocolo denominou "commit quorum" e a mensagem "commit" pode ser difundida. Isto significa que as mensagens de MCSL de MRB devem ser colocadas em MRQ na ordem indicada por MCSL e o COML deve ser atualizado em todas MSP's. Em caso contrário, o iniciador difunde um "abort" para todos MSP's.

Condição de Terminação

Se algum MSP, denominado MSP_j, não receber nenhuma mensagem de decisão (commit ou abort) até um timeout, este entra em operação de terminação para tentar engajar ou abortar, sem a mensagem de decisão do iniciador. Para isto o mesmo deve conhecer os MSP's escravos e cooperar com eles para a terminação da execução do protocolo. Detalhes específicos da terminação são apresentados em [Luan 90].

2.3.5 - Protocolo de Difusão Confiável de Melliar-Smith

O protocolo apresentado em [Melliar-Smith 90] é da classe dos protocolos com difusão totalmente ordenada (difusão atômica). Ele apresenta características de protocolo assíncrono suportando as propriedades de acordo, de ordenação e de terminação na presença de faltas temporais.

Este protocolo constitui-se, básicamente, de dois outros protocolos:

- O **Trans Protocol** que garante a propriedade de acordo.
- O **Total Protocol** que com alta probabilidade de acerto faz o ordenação total das mensagens difundidas garantindo a propriedade de ordenação.

O Trans Protocol leva o reconhecimento das antigas mensagens difundidas nas novas. Uma vez que a mensagem foi reconhecida por uma segunda estação, uma terceira não necessita reconhecê-la novamente bastando apenas reconhecer a mensagem da segunda. No caso de uma quarta estação não ter recebido a mensagem da primeira, a segunda estação alerta esta estação da perda. Assim esta quarta estação inclui um reconhecimento negativo da mensagem da primeira estação em sua próxima mensagem.

O reconhecimento positivo e negativo contido nas mensagens permite às estações determinarem se uma estação recebeu uma mensagem mesmo que os reconhecimentos de mensagem não sejam recebidos diretamente. Isto garante que todos os processos construam uma mesma ordem parcial das mensagens difundidas.

A ordenação total não ocorre imediatamente após a mensagem ter sido difundida, mas espera pela recepção de mensagens de outras estações trazendo o "acknowledgement". Por isto, para evitar grande

latência, um processo que não tenha mensagens pendentes deve construir uma mensagem nula para levar seus reconhecimentos.

Em [Melliar-Smith 90] são definidos os termos:

- **"Observable Predicate for Delivery"** denotado por **OPD(P,A,C)** que especifica que umprocesso P pode ter certeza que um processo P_c recebeu e reconheceu, direta ou indiretamente, a mensagem A quando enviou a mensagem C.

- "Mensagem C **segue** a mensagem B", isto é denominado se, e somente se, OPD(P,B,C) e, para toda e qualquer mensagem A se OPD(P,A,B) implicar em OPD(P,A,C).

- **"Mensagem Candidata"** denotado para toda e qualquer mensagem que não segue, na ordem parcial, nenhuma outra mensagem além das que já estão na ordem total.

O Total Protocol faz a ordenação total anexando uma mensagem candidata (ou um conjunto de candidatas) se existir um número mínimo de mensagens de estações distintas, que seguem a esta, e somente a esta, mensagem candidata (ou conjunto de candidatas). Este número mínimo é dado pelos parâmetros N_d e N_v do algoritmo. Desta forma, a decisão de anexar uma candidata é determinada com base nesses parâmetros e na ordem parcial das mensagens, e não pela decisão em conjunto com outras estações (coordenação distribuida).

2.4 - Comentário dos protocolos descritos

O trabalho apresentado em [Chang 84] só permite que uma mensagem seja engajada após (L+1) transferências de token (para um sistema L-resistente) e descartada após o token ter circulado uma vez pelo anel. Com isto a latência da mensagem na camada de difusão e seu custo de armazenagem são ditadas pelo tempo máximo de permanência do token em uma estação e o tamanho da token list. Se o token for passado rapidamente o custo de latência e de armazenagem serão pequenos, porém aumenta o custo de comunicação e vice-versa. Sua abordagem anel é muito sensitiva a estações faltosas o que não é bom quando a taxa de estações faltosas não é desprezível em relação à taxa de transferência de token, já que isto provoca frequentes invocações do protocolo de reformação, aumentando significativamente o overhead e introduzindo longos delays.

O ABCAST de Birman requer 2N mensagens-protocolo por mensagem-difundida em operações sob condições normais (sem falha). As mensagens-difundidas, após receberem seu "timestamp final" devem esperar até que todas as mensagens de timestamp menor, presentes no "delivery queue", tenham sido engajadas. Os protocolos apresentados por Birman, como o de Chang, não suportam particionamento da rede.

Uma abordagem totalmente diferente é apresentada por Melliar-Smith onde é assumida uma ordenação total de alta probabilidade de acerto. No entanto, existem chances destas mensagens não serem totalmente ordenadas [Kaashoek 89], visto que o protocolo não requer detecção de estações faltosas. Neste protocolo, poucas ou nenhuma mensagem-protocolo são utilizadas mas, as mensagens não podem ser engajadas pela aplicação até que muitas outras mensagens difundidas tenham sido recebidas. Assim o

custo de latência depende da velocidade com que as novas mensagens, enviando ack's das antigas, cheguem as estações.

O protocolo de Luan e Gligor, sob operações em condições normais, requer três fases para se completar e aproximadamente 4N mensagens-protocolo. Se mais de 4N mensagens são difundidas numa execução do protocolo, o overhead se distribui entre estas, vindo a ser bem baixo o número de mensagens de controle por mensagens difundidas. A latência do algoritmo, no entanto, independe do número de mensagens-protocolo.

Cristian apresenta um protocolo síncrono com custo para com mensagens de controle bem reduzido (nulo). Entretanto o tempo para o engajamento das mensagens é fixo e baseado no comportamento de pior caso. Tal tempo é determinado pelo limite superior da exatidão (ε) entre os relógios sincronizados e tempo máximo de difusão das mensagens entre todas estações. Os custos involvidos com a sincronização dos relógios também tem de ser considerados uma vez que não são desprezíveis.

A tabela 2.1 apresenta a síntese das principais características dos protocolos apresentados.

	BIRMAN	CHANG	CRISTIAN	LUAN	MELLIAR-SMITH
Terminação	Assíncrona	Assíncrona	Fortemente Síncrona	Assíncrona	Assíncrona
Ordenação do Protocolo	Totalmente Ordenada	Totalmente Ordenada	Totalmente Ordenada	Totalmente Ordenada	Totalmente Ordenada
Acordo/ coordenação	Distribuído	Centralizado	Distribuído	Centralizado	Distribuído
Faltas Suportadas	Faltas de "crash"	Faltas por Omissão	[omissão,...., maliciosas]	Faltas por omissão	Faltas Temporais
Overhead de mensagens	2 N	[N, 2N]	0[1]	(0,4N]	Variável (depende dos ACK's nulos)
Latência	Após 2 rounds de troca de mensagens	L transferências de token, para um sist. L-resistente	Após um delay pessimista (Δ) fixo	Duas execuções de protocolo[2]	Aleatório[3]
Particiona-mento da rede	Não admissível	Admite engajamento de mensagens	Não admissível	Admite engajamento de mensagens	Admite engajamento de mensagens

1 - O custo de sincronização deve ser considerado
2 - Uma execução de protocolo envolve espera das mensagens e três rounds de troca de mensagens
3 - Depende da existência de um número mínimo de mensagens de estações distintas

Tabela 2.1 - Síntese das principais características dos Protocolos de Difusão Confiável

2.5 - Conclusões

Neste capítulo foi identificado o domínio que a dissertação se propõe entrar: a necessidade de um serviço de comunicação eficiente, que garanta a recepção de mensagens, mesmo na presença de faltas, por todas as estações corretas do grupo. Tal serviço foi definido como "**Protocolo de difusão confiável**".

Foram introduzidas as propriedades, sobre as quais se fundamentam os protocolos de difusão confiável: acordo, ordenação e terminação. Segundo os graus de atendimento a estas propriedades foram estabelecidas classificações dos protocolos de difusão.

Por fim, foram apresentados exemplos significativos de protocolo de difusão confiável, presentes na literatura. Os estudos e conclusões deste capítulo serviram de base para a proposição do modelo e do algoritmo discutidos nos próximos capítulos.

CAPÍTULO 3

PROPOSTA DE PROTOCOLO DE DIFUSÃO CONFIÁVEL

3.1 - Introdução

Este capítulo tem como objetivo principal a apresentação de uma proposta de protocolo de difusão confiável. Os objetivos que foram fixados para este algoritmo são: ordenação total, baixo "overhead" e latência limitada. Os aspectos de acordo devem ser mantidos mesmo em presença de faltas de omissão e crash. O algoritmo deve possuir um grau de "recuperação" ("Resiliency") que sempre permite a um participante da comunicação recuperar mensagens já engajadas em "commits" anteriores.

Neste capítulo é apresentado inicialmente um modelo geral de protocolos de difusão onde são salientados os principais aspectos deste tipo de comunicação. O algoritmo proposto é então introduzido com algumas definições necessárias para a compreensão do mesmo, seguido da descrição do algoritmo em funcionamento normal e dos tratadores de exceções.

3.2 - Modelo Geral

3.2.1 - Propriedades básicas de um protocolo de difusão

Como foi salientado no capítulo anterior, um protocolo de difusão confiável esta fundamentado nas três propriedades básicas [Cristian 85]: acordo, ordenação e terminação. As propriedades de terminação e acordo devem ser observadas em todos os protocolos de difusão confiável, ou seja: uma mensagem difundida ou é aceita por todos os participantes operacionais, ou por nenhum destes e ainda, que esta aceitação ou não da mensagem ocorra dentro de um tempo limitado. As necessidades do processamento distribuído devem definir que tipo de ordenação deve estar presente no protocolo de difusão.

Com base nisto, os elementos essenciais ao projeto de serviços de difusão confiável são protocolos de acordo, algoritmos distribuídos de detecção de componentes faltosos e mecanismos para implementação

de ordenação. Para a elaboração ou escolha destes algoritmos e mecanismos podem ser utilizados alguns parâmetros ligados, por exemplo, ao aspecto de custo (ou de desempenho).

Os custos de um protocolo de comunicação são basicamente três [Segall 83]:

- **Custo de comunicação**: quantidade de mensagens-protocolo (e/ou mensagens maiores) necessária para realizar a difusão de uma mensagem da aplicação (denominado neste trabalho de overhead).
- **Latência**: Demora vista pelo emissor, entre a difusão da mensagem e o seu engajamento.
- **Requisitos de recursos**: necessidade de recursos básicos (memória) de qualquer algoritmo de comunicação.

As metas de desempenho em um protocolo são as de minimizar o custo de overhead e de latência em condições normais de operação.

3.2.2 - Modelo Geral de Protocolos de difusão confiável

Um sistema distribuído típico, conforme mostra a figura 3.1, consiste de um conjunto de estações interconectadas por uma rede de comunicação. Em cada estação pode existir um ou mais processos se executando, no entanto, por uma questão de simplicidade, será assumida a existência de apenas um processo por estação.

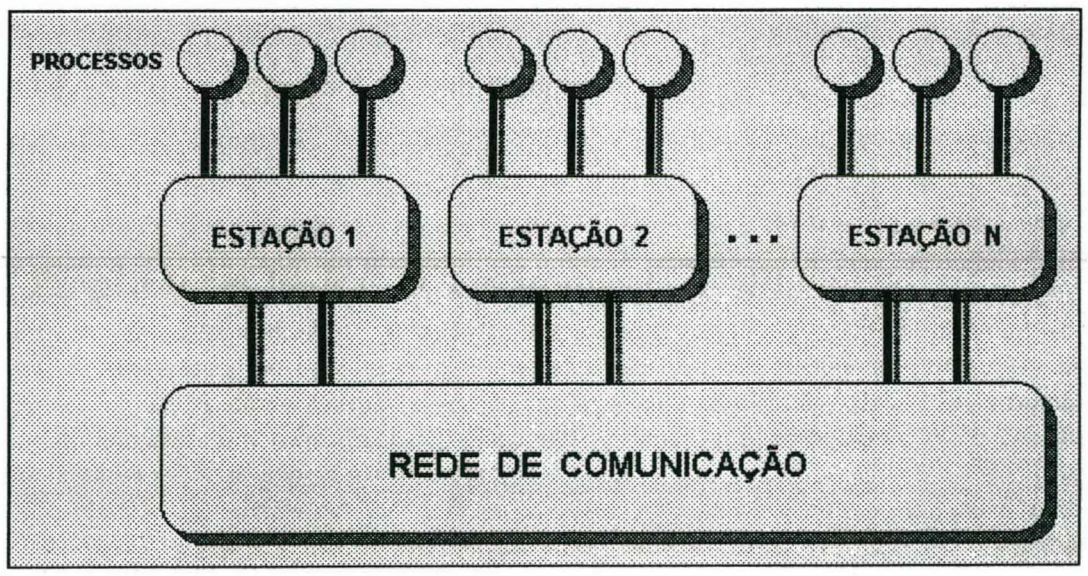


Figura 3.1 - Modelo do sistema

Como não há compartilhamento de memória entre as estações, a única forma de comunicação é através da rede. O Protocolo de Difusão é um serviço que opera numa camada denominada "Camada de Difusão Confiável" e que se localiza entre a camada de aplicação e o suporte de comunicação remota do sistema (figura 3.2).

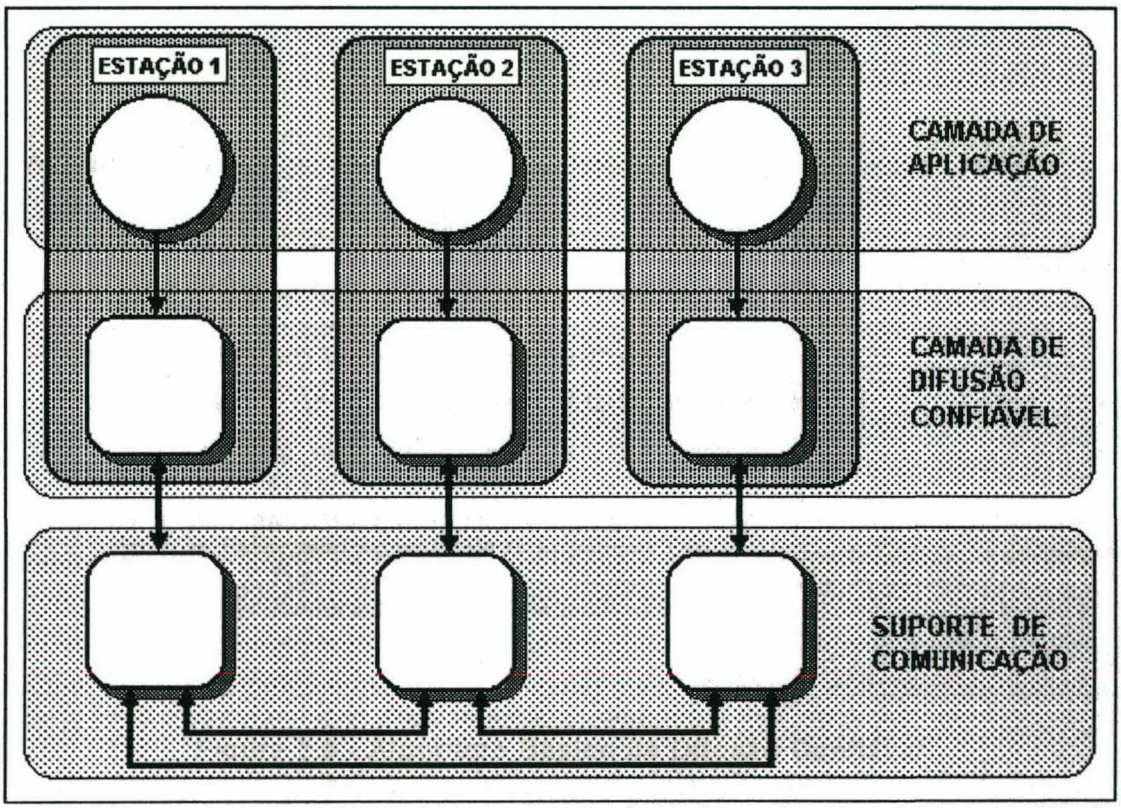


Figura 3.2 - Sub-sistema de comunicação

O suporte de comunicação remota contém todo o hardware e software necessários para habilitar uma mensagem a ser enviada de um processo para outro. A transmissão de uma mensagem pelo suporte de comunicação (transmissão física na rede) pode se dar através de serviços de comunicação ponto a ponto ou de comunicação multiponto (difusão). Toda mensagem que chega à camada de difusão passou pelos testes de integridade do suporte de comunicação (por exemplo, testes de CRC). Os serviços do suporte de comunicação, como é usual, fazem descarte de mensagens com erros e, no caso de serviço ponto a ponto apresenta um certo nível de recuperação de erros quando opera no estilo circuito virtual (serviço datagrama só detecta erro, não recupera).

A camada de difusão confiável implementa a facilidade de envio confiável de uma mensagem de um processo qualquer para um grupo conhecido de processos. Este serviço está fundamentado sobre o serviço de comunicação multiponto (inconfiável) do suporte de comunicação. O envio de uma mensagem inclui, em geral, outras informações que devem ser utilizadas pela camada de difusão nas outras estações. Dependendo do protocolo de difusão que está sendo executado, pode haver mais rodadas de comunicação entre as estações a nível da camada de difusão no sentido de garantir todas as propriedades básicas do

protocolo. O engajamento das mensagens se dará com a verificação das propriedades e a consequente passagem destas do serviço de difusão confiável para a camada de aplicação.

O protocolo executado pela camada de difusão depende do nível de tolerância à faltas fornecido e da maneira que as mensagens são ordenadas.

```
Protocolo Difusão Confiável :: X

Aplicação :: APj

Camada de Difusão Confiável na estação i :: CD(i)

Mensagem enviada pela Aplicação  para difusão :: msg

Mensagem difundida pela Camada de Difusão Confiável :: mdif


X:: * [     [  APj?msg  mdif := msg + mensagens de controle;
               (i:1..N) CD(i);  CD(i)!mdif                                    ]
        [] [  (i:1..N) CD(i);  CD(i)?mdif  <  Ordena mdif no buffer de recepção >   ]
        [] [  < Existe mdif no buffer de recepção da CD(j) >;
              * [  < Protocolo de acordo>;   <Reordena mdif no buffer de recepção>]
              * [  < Aceita mdif >;   < Envia mdif aceita para a APj >      ]    ] ]
```

Figura 3.3 - Algoritmo Básico de Difusão Confiável

Em resumo, uma execução do protocolo de difusão confiável, que se apresenta esboçado em pseudo-linguagem; onde tem uma parte de CSP e parte em lingugem natural; na figura 3.3, se dá em três fases distintas:

Fase 1: Difusão da mensagem a todos processos

Fase 2: Recepção da mensagem pelos processos com a
Verificação das propriedades de acordo, ordenação e terminação

Fase 3: Aceitação da mensagem pela camada de difusão e, consequentemente, seu engajamento

As diferenças entre os algoritmos estão nas diferentes técnicas utilizadas nestas três fases.

3.3 - Algoritmo Proposto

O protocolo proposto, como os demais, está localizado numa camada acima do suporte de comunicação, o qual fornece, além do serviço ponto a ponto, a transferência multiponto de mensagens (difusão de mensagens), sendo que os serviços de comunicação multiponto da rede não são confiáveis. A estratificação do sistema é a mesma já apresentada na figura 3.2.

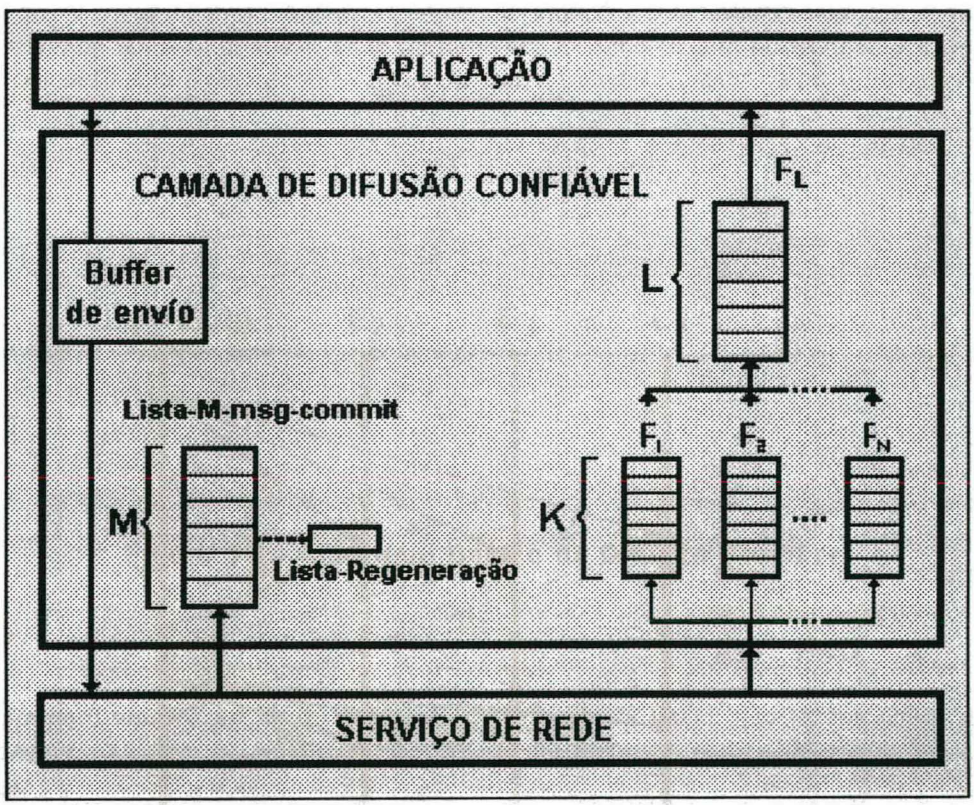


Figura 3.4 - Estratificação do protocolo proposto

Um processo na aplicação, ao ativar a primitiva de envio em comunicação de grupo, tem a sua mensagem emitida pela camada de difusão confiável, para os participantes do grupo especificado. A mensagem difundida apresenta o número da estação emissora e o número da mensagem por ela emitida, como informações de controle acrescentadas pela camada de difusão confiável na estação emissora. O número da mensagem emitida é que garante a propriedade de ordenação local das mensagens.

Na recepção das mensagens difundidas na rede, é verificada a estação emissora e sua ordem local. Diante destas informações, a mensagem recebida na camada de difusão confiável é colocada numa das N filas de mensagens não engajadas (F_i) (secção 3.3.1) presentes na estação, correspondente ao

emissor E_i. A figura 3.4 mostra as filas e buffers presentes na camada de difusão confiável de cada estação. Nesta figura é salientado que a capacidade de recepção de mensagens difundidas e ainda não engajadas, da estação, é de K mensagens para cada um dos N potenciais emissores no grupo.

Este protocolo, a exemplo de [Chang 84] e [Luan 90], apresenta o processo de engajamento centralizado sobre uma estação: a coordenadora. O papel de coordenadora é transferido, através de passagem de token, entre as estações participantes do grupo, no fim de cada engajamento. Neste sentido, as estações se apresentam ordenadas em um anel virtual segundo seus endereços lógicos (i= 1,2, ... , N).

Ao se completar L mensagens difundidas, presentes entre as filas de mensagens não engajadas - F_i (ou ainda por decurso de prazo) a estação coordenadora envia uma mensagem-protocolo, "pedido-de-commit", às demais estações. Com tal pedido, inicia-se uma "execução de engajamento", afim de engajar as L (ou menos) mensagens presentes nas Filas de mensagens não engajadas (F_i) e transferir o token a uma "nova coordenadora". Assim, se as demais estações concordarem com a lista de mensagens a engajar e estiverem aptas ao commit, estas enviarão um reconhecimento positivo ao pedido-de-commit. No caso de desacordo com a lista de mensagens ou com a transferência do token, o reconhecimento deve ser negativo.

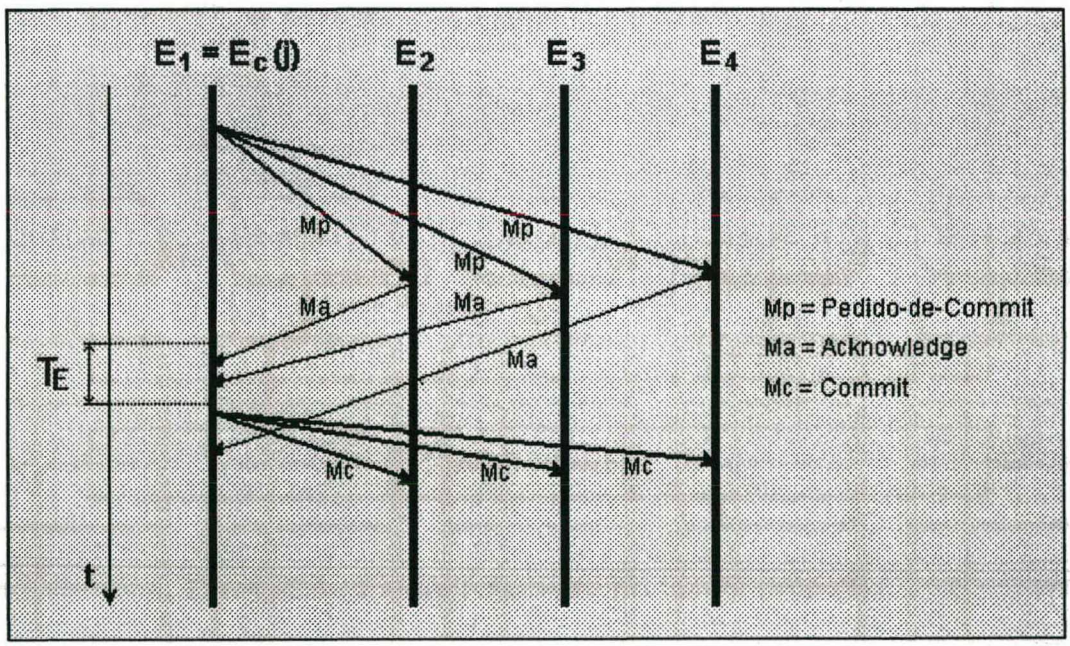


Figura 3.5 - Execução de Engajamento

Se a maioria das estações enviaram mensagem de reconhecimento, com nenhum reconhecimento negativo ao pedido-de-commit, a estação coordenadora deve transferir o token a uma nova coordenadora difundindo a mensagem-protocolo "commit". Com a recepção desta mensagem-protocolo "commit" as mensagens são engajadas, estações faltosas retiradas do grupo de estações participantes, a mensagem commit armazenada na Lista-M-msg-commit (secção 3.3.1) e a nova coordenadora assume a condição de coordenadora atual finalizando a execução de engajamento e a execução de protocolo j. Uma

"execução de protocolo" está portanto, relacionada com o preenchimento da capacidade de engajamento (L) da camada e sua execução de engajamento, devendo iniciar com o recebimento do token por uma estação e finalizar com a transferência deste a outra estação do grupo. A figura 3.5 sintetiza os passos para o engajamento de mensagens.

A ordenação das mensagens a engajar será fila por fila em ordem crescente das "estações-L", isto é, das estações situadas entre a coordenadora e a nova coordenadora, incluindo estas, que possuem mensagens a engajar. A figura 3.6 mostra esta ordenação. A figura 3.6a apresenta as N F_i (Filas de mensagens não engajadas) presentes em uma estação do grupo antes do engajamento. As estações-L da figura são as estações E_{i-1}, E_i e E_{i+1} sendo que as L mensagens difundidas se completam com a terceira mensagem de F_{i+1} (Fila de mensagens não engajadas referente à estação E_{i+1}). Após a aceitação destas mensagens (engajamento), E_{i+1} torna-se a nova coordenadora e as mensagens são retiradas, em todas as estações do grupo, de suas respectivas Filas na ordem de engajamento e colocadas na F_L, como apresentado na figura 3.6b. A figura 3.6c apresenta as N F_i desta estação do grupo após o engajamento destas mensagens.

Afim de se evitar alto custo de "overhead" devido a frequentes reconfigurações e de tratamento de faltas, o protocolo só considera uma estação "faltosa" após M ausências consecutivas de execuções de protocolos, isto é, o "crash" da estação é caracterizado por M faltas de omissão consecutivas [Nacamura 92]. Assim, se uma estação não participa de (M - 1) execuções de protocolo, na M-ésima execução a coordenadora difunde, na mensagem-protocolo "commit", o nome da estação como sendo faltosa. A partir daí, nenhuma mensagem desta estação poderá ser recebida antes de sua reinserção.

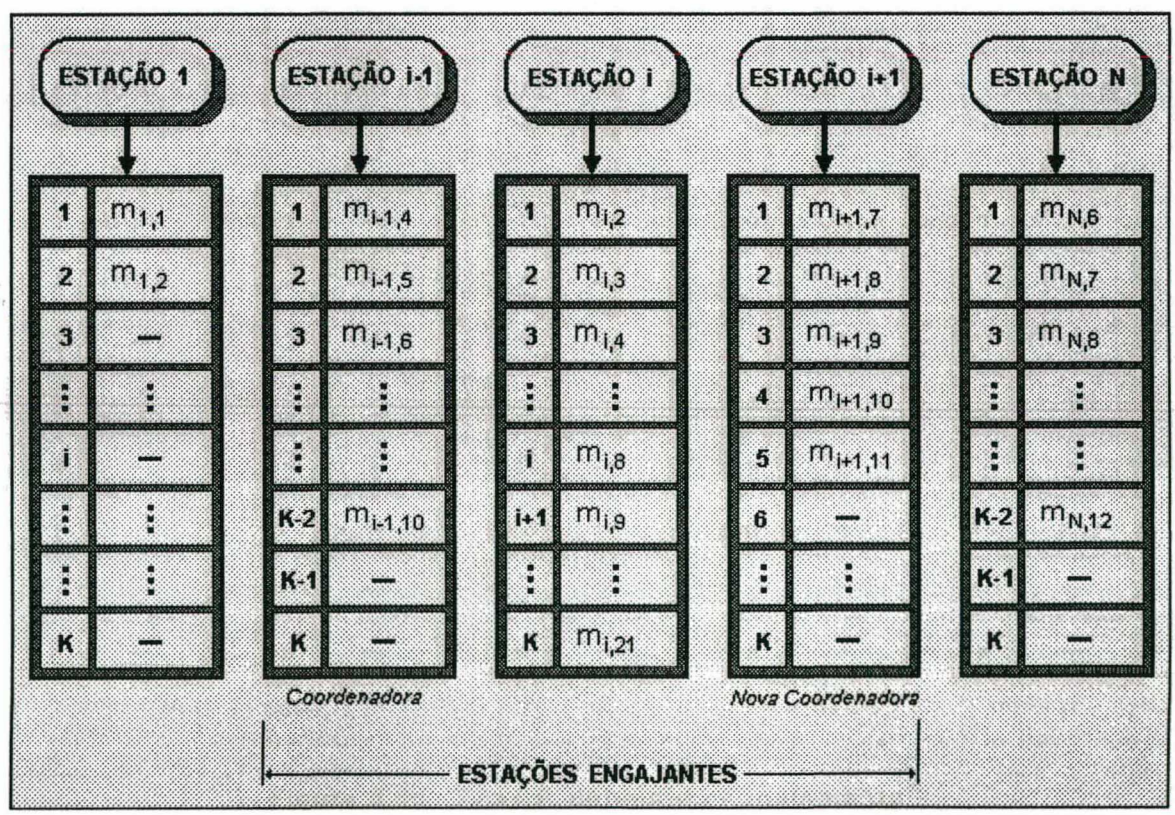


Figura 3.6a - Filas F_i antes da execução de engajamento

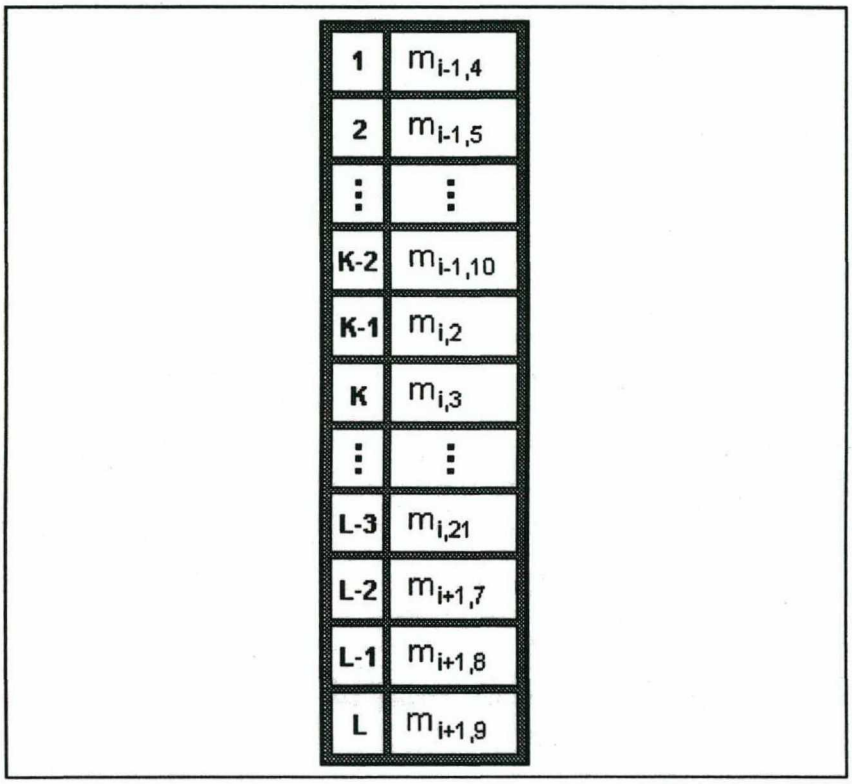


Figura 3.6b - Fila F_L (Mensagens que serão engajadas)

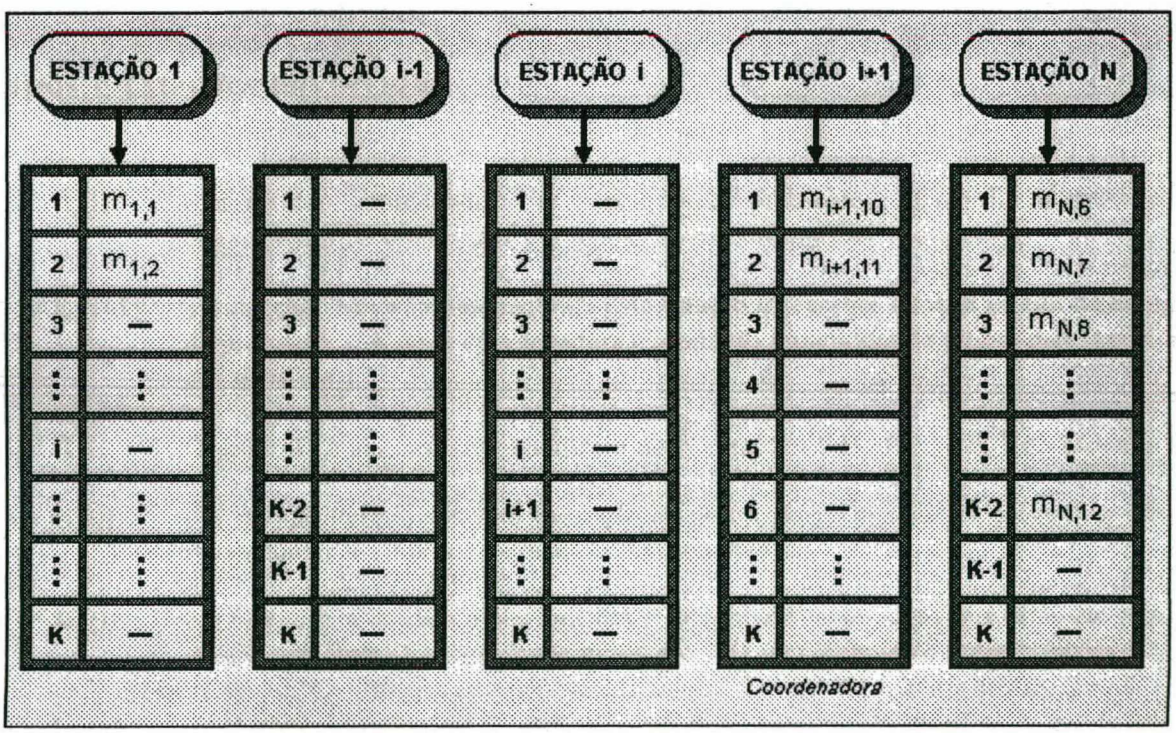


Figura 3.6c - Filas F_i depois do engajamento

3.3.1 - Definições

Neste subitem são introduzidas as principais definições e notações que serão utilizadas no restante deste trabalho:

Parâmetros Básicos

N: número de estações em um grupo G.

M: número máximo de vezes consecutivas que uma estação pode não participar de execuções de protocolo sem ser considerada faltosa (Grau de omissão).

K: capacidade máxima de recepção de mensagens, ainda sem engajamento, provenientes de um emissor E_i.

R: número máximo de estações que podem faltar numa execução de protocolo sem que o sistema perca suas propriedades de regeneração de mensagens perdidas (grau de "robustez").

L: capacidade máxima de engajamento de mensagens na camada de difusão confiável em uma execução do protocolo.

Grupos e estações envolvidas

$E_c(j)$: é a estação coordenadora na execução de protocolo j. Controla o processo de engajamento de mensagens e a reposição das mesmas na execução j.

$E_c(j+1)$: é a estação "Nova Coordenadora". Esta estação irá receber o token no fim da execução de protocolo j. A nova coordenadora é a estação possuidora da última mensagem a ser engajada em j.

G : O grupo G é um conjunto composto de N estações $\{ E_1, E_2, .. , E_N\}$ do sistema S.

$G_p(j)$: é o Grupo-Participantes na execução de protocolo j. Ou seja, estações consideradas operacionais no inicio da execução de protocolo j.

$G_L(j)$: é o Grupo-L. Formado pelas estações E_i origem das mensagens que estão em processo de engajamento na execução de protocolo j. $G_L(j)$ é composto pelas estações-L.

$G_A(j)$: é o Grupo-Ausentes na execução j, isto é, o conjunto de estações E_i pertencentes a $G_p(j)$ que se "omitiram" durante a execução de protocolo j.

$G_r(j)$: Grupo-Recuperando a execução j é o conjunto de estações E_i pertencentes a $G_A(j)$ que estão em processo de recuperação da execução de protocolo j.

$G_f(j)$: é o Grupo-Falho na execução de protocolo j, formado pelas estações E_i de $G_A(j)$ consideradas em situação de "crash" no final da execução de protocolo j:

$$G_f(i) = \bigcap_{m=j-M+1}^{m=j} \left(G_A(m) - G_r(m)\right)$$

$G_i(j)$: o Grupo-Inserido corresponde ao conjunto de estações inseridas em $G_p(j)$ antes do inicio da execução j.

$G_F(j)$: o Grupo-Falho em G é formado pelas estações de G consideradas falhas em todas as execuções do protocolo até j e que não foram reinseridas em $G_p(j)$:

$$G_F(i) = \bigcup_{m=1}^{m=j} \left(G_f(m-1) - G_i(m)\right)$$

$G_{EG}(j)$: é o Grupo-Engajante formado pelo conjunto de estações E_i de $G_p(j)$ que reconheceram ao pedido-de-commit da execução de protocolo j e que supostamente, engajaram as mensagens propostas desta execução.

Mensagens, Listas e Buffers

$m_{i,k}$: mensagem difundida pela estação E_i, com o número de sequência k. O número de sequência determina a ordem (local) de emissão.

Lista-Regeneração :

lista composta pelas mensagens engajadas num determinado "commit". Para cada commit existirão (R+1) estações com as mensagens correspondentes.

Lista-M-msg-commit :

Lista com as M últimas mensagens de commit executados pelo protocolo. Presente em todas estações participantes.

Fila de mensagens não engajadas, F_i :

corresponde às filas de mensagens difundidas, porém, ainda não engajadas pela camada de difusão confiável. Cada estação emissora E_i de G tem uma fila F_i, presente em todas as estações de $G_p(j)$, para bufferização de suas mensagens emitidas. Cada fila F_i limita em K a capacidade de emissão de E_i entre engajamentos de suas mensagens.

Fila de mensagens engajadas, F_L:

corresponde à fila de mensagens que estarão disponíveis a camada de aplicação, já engajadas. A capacidade de engajamento do protocolo na execução j é definida pelo tamanho deste buffer (L).

Ponteiros

P_{Fi}: é o ponteiro da fila F_i, indicando a próxima mensagem esperada em Fi, referente a emissora E_i.

$P_{EGi}(j)$: é o ponteiro da última mensagem a ser engajada em F_i, durante a execução de protocolo j. E_i pertence a $G_L(j)$.

$L_{me}(j)$: corresponde à lista dos ponteiros $P_{EGi}(j)$ indicando, na execução de protocolo j, quais as últimas mensagens nas F_i das estações E_i em $G_L(j)$, que terão suas mensagens engajadas:

$$L_{me}(j) = \{ P_{EG_i}(j) \mid E_i \in G_L(j) \wedge (i = 1, 2, \ldots, N) \}$$

Tempos

T_E: Tempo máximo de espera, pela coordenadora, pelos reconhecimentos ao pedido-de-commit.

T_A: Timeout para a ativação de uma execução de engajamento.

T_{reconf}: Timeout para a circulação total do token pelo anel virtual.

Diante das definições que foram introduzidas neste item pode-se estabelecer algumas relações:

$$G = (G_p(j) \cup G_F(j)) \tag{3.1}$$

$$G_p(j) = (G_A(j) \cup G_{EG}(j)) \tag{3.2}$$

$$G_L(j) \subset (G_A(j) \cup G_{EG}(j)) \tag{3.3}$$

3.3.2 - Protocolo de difusão confiável

Uma estação E_i pertencente a $G_p(j)$ pode enviar ou receber, sobre o suporte de comunicação, as mensagens da aplicação independente do andamento da execução de engajamento j (engajamento de mensagens na camada de difusão confiável). Neste sentido, estes três aspectos do Algoritmo: o envio e a recepção de mensagens da aplicação sobre a rede e o engajamento, serão apresentados separadamente a seguir.

3.3.2.1 - Difusão de uma mensagem sobre o suporte de comunicação

Toda mensagem vinda da aplicação, na camada de difusão confiável, é acrescida das informações identificando a mensagem e sua estação emissora. A camada de difusão confiável, na estação E_i difunde, então, a mensagem ($m_{i,k}$) com suas respectivas informações de controle, em $G_p(j)$, usando os serviços de rede.

A estação E_i pode emitir mensagens de sua aplicação em $G_p(j)$ em qualquer instante da execução de protocolo j. A única limitação imposta é que o número de mensagens não engajadas presentes na camada de difusão confiável, referentes a emissões de E_i, não ultrapassem o valor K.

A estação E_i é responsável pela retransmissão das mensagens que tiveram origem nesta estação, desde que estas mensagens não tenham sido engajadas ainda.

3.3.2.2 - Recepção de uma mensagem do suporte de comunicação

Na recepção de uma mensagem $m_{i,k}$ difundida sobre a rede, as informações de controle são verificadas, de modo que a mensagem possa ser colocada na fila F_i correspondente a sua estação emissora E_i, na ordem do seu número de sequência k (ordem de emissão da sua estação de origem). Esta mensagem então, passará a fazer parte das mensagens não engajadas.

As mensagens duplicadas são descartadas; o controle de identificação está baseado no número de sequência k da estação emissora. Uma mensagem duplicada é detectada quando o número da sequência recebida é menor que o número esperado para a emissora considerada. A perda de mensagem tem sua detecção quando uma mensagem apresenta o número de sequência maior do que o número esperado para o emissor em questão. A perda, pelo receptor, da sequência de numeração nas mensagens recebidas referente a um emissor E_i, faz com que a estação receptora envie a E_i um pedido de retransmissão da(s) mensagem(s) cujo número de identificação está em falta em sua fila F_i.

3.3.2.3 - Execução de Engajamento

Conforme descrito anteriormente, o protocolo apresenta uma capacidade de engajamento de L mensagens por execução do protocolo. As mensagens para a aplicação devem se acumular na camada de difusão confiável, nos buffers de mensagens não engajadas (N F_i), até atingir o valor L de mensagens ou

ainda expirar o Timeout de ativação da execução de engajamento (T_A). Em uma destas situações, a estação coordenadora de $G_p(j)$, a $E_c(j)$, deve iniciar o processo de engajamento destas mensagens, enviando um "pedido-de-commit", o qual da inicio a execução de engajamento j.

Mensagens a engajar, ordenação e próxima coordenadora na execução j

As mensagens que serão negociadas e que deverão ser engajadas no final da execução j, são as presentes nas filas F_i dos emissores E_i pertencentes a $G_L(j)$. A definição dos elementos de $G_L(j)$ depende das mensagens presentes nas filas Fi na camada de difusão.

Neste sentido, é definido $B_i(j)$ como a "bufferização j", presente em F_i no inicio do engajamento j, ou seja, o conjunto de mensagens provenientes de E_i, armazenadas em F_i, segundo a ordem local definida pelos números de sequência das mensagens:

$$\mathbf{B_i(j) = \{ m_{i,k}, m_{i,k+1}, \ldots, m_{i,k+s} \} \wedge}$$

$$\mathbf{m_{i,k} \rightarrow m_{i,k+1} \rightarrow \ldots \rightarrow m_{i,k+s}} \qquad \text{(ordem local)}$$

onde a relação, notada por "$\rightarrow$", indica precedência com base no número de sequência das mensagens (k).

Considerando a estação E_i como a coordenadora da execução j ($E_i = E_c(j)$), e λ a distância da nova coordenadora ($E_c(j+1)$) em relação a E_i na ordenação do anel, tem-se então:

$$\sum_{m=i}^{m=\lambda} card(B_m(j)) \leq L \qquad (3.4)$$

onde $card(B_i(j))$ é a função que indica a cardinalidade do conjunto $B_i(j)$ e $1 \leq \lambda \leq N$.

A bufferização $B_L(j)$ define o conjunto das mensagens engajadas no final da execução de protocolo j e presentes em F_L. O valor L é a cardinalidade máxima que $B_L(j)$ pode alcançar (na situação sem esgotamento do timeout T_A).

As mensagens são ordenadas em $B_L(j)$ segundo uma ordem total tomando como base as ordens locais em cada $B_i(j)$ e a ordem estabelecida entre as bufferizações no engajamento j:

$$\mathbf{B_i(j) \rightarrow\rightarrow B_{i+1}(j) \rightarrow\rightarrow \ldots \rightarrow\rightarrow B_{i+\lambda}(j)}$$

onde a relação notada por "$\rightarrow\rightarrow$", indica a precedência com base no número de sequência (i) das estações no anel virtual.

Assim, duas mensagens engajadas em j, $m_{i,k}$ e $m_{s,r}$, apresentarão a relação de precedência:

$$\mathbf{m_{i,k} \Rightarrow m_{s,r}}$$

indicando que $m_{i,k}$ precede $m_{s,r,}$ segundo a ordem total estrita, notada por "⇒", se:

OU

$$B_i(j) \rightarrow\rightarrow B_s(j) \text{ onde } m_{i,k} \in B_i(j),\ m_{s,r} \in B_s(j),\ \text{para } i < s$$

OU

$$m_{i,k} \rightarrow m_{s,r} \qquad \text{onde } m_{i,k},\ m_{s,r} \in B_i(j),\ \text{para } i = s \wedge k < r$$

Algumas considerações podem ser tiradas destas relações:

1) A nova coordenadora será a estação que tiver a última mensagem em $B_L(j)$, segundo a ordenação "⇒" e mantendo $card(B_L(j)) \leq L$.

2) Toda mensagem que chegue em F_i após o inicio da execução de engajamento j, pertence a $B_i(j+1)$.

3) Toda mensagem pertencente a fila da nova coordenadora (E_c (j+1)) que não entrar na bufferização $B_L(j)$, no final da execução de protocolo j, não será considerada pertencente a $B_{i+\lambda}(j)$, mas sim $B_{i+\lambda}(j+1)$

Ativação da execução de engajamento j

A coordenadora $E_c(j)$ inicia a execução de engajamento j quando L mensagens estão presentes nas filas F_i ou quando do esgotamento do decurso de prazo (timeout T_A) para a ativação da execução de engajamento j. A estação $E_c(j)$ deve então enviar um "**pedido-de-commit(j)**". Esta mensagem de protocolo contem informações tais como: o número da versão atual do token e a lista $L_{me}(j)$ com as últimas mensagens a engajar em cada fila F_i das estações E_i pertencentes a $G_L(j)$ e ainda, a informação sobre a próxima coordenadora.

Processamento do pedido-de-commit(j)

Toda estação ao receber o pedido-de-commit(j) deve verificar as seguintes condições:

C1 - $L_{me}(j)$ produzida por $E_c(j)$ é correta.

$L_{me}(j)$ é considerada correta se a proposição de ponteiros $P_{EGi}(j)$ (ponteiros indicando a última mensagem a ser engajada em F_i, na execução j) forem considerados "apropriados", ou seja, não excluirem mensagens de F_i que num engajamento anterior, onde E_i pertencia ao grupo de estações-L, não foram consideradas.

O mecanismo de circulação de token determina a circulação do papel de coordenadora nos engajamentos de mensagens mas também, corresponde a circulação do status de estação em $G_L(j)$.

Considerando então a execução de protocolo m, onde m < j (m antecede j), tal que:

$$E_i \in \{G_L(m) \cap G_L(j)\} \wedge$$
$$E_i \notin \{G_L(m+1) \cup G_L(m+2) \cup \ldots \cup G_L(j-1)\}$$

neste caso, o ponteiro $P_{EGi}(j)$ é dito **"Apropriado"** se:

$$P_{EGi}(j) > P_{EGi}(m) \quad \vee$$
$$P_{EGi}(j) = SM$$

onde SM corresponde a um valor indicando "sem mensagem".

A lista $L_{me}(j)$ que apresentar $P_{EGi}(j)$ "apropriados" para todas as filas F_i de estações $E_i \in G_L(j)$, será considerada correta.

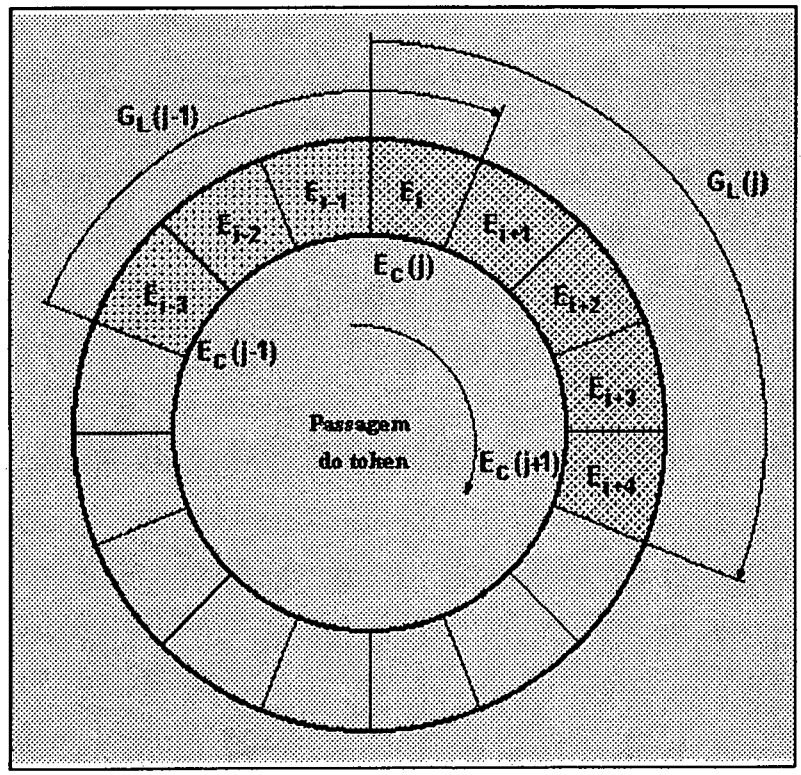


Figura 3.7 - Passagem de Token

Na situação considerada, para $m = j\text{-}1$ e $E_i \in \{G_L(m) \cap G_L(j)\}$ tem-se E_i participando, em duas execuções de protocolos consecutivas, das estações-L o que implica que E_i obrigatoriamente deve ser a $E_c(j)$. Em casos de $m \neq j\text{-}1$, duas participações consecutivas de E_i como estações-L ($E_i \subset G_L(j) \wedge E_i \subset G_L(m)$) corresponde a uma circulação completa do token. A figura 3.7 sintetiza o mecanismo de passagem do token.

Apartir das considerações acima, pode-se afirmar que as mensagens difundidas serão engajadas no máximo em uma circulação completa do token.

C2 - Toda estação E_s que receber o pedido-de-commit(j) deve estar apta a participar da Execução de engajamento j.

Uma estação E_s estará apta a participar de execução de engajamento j se $E_s \in G_p(j) \wedge E_s \notin G_A(j\text{-}1)$. Se E_s esteve "ausente" na execução j-1 ($E_s \in G_A(j\text{-}1)$), não poderá se manifestar ao pedido-de-commit(j) antes de recuperar os commits anteriores que tenha perdido.

Não basta $E_s \in G_p(j) \wedge E_s \notin G_A(j\text{-}1)$ para estar apta para a execução j. É necessário que apresente a mesma imagem das mensagens nas filas F_i (as mesmas bufferizações $B_i(j)$) que a $E_c(j)$ está propondo para o engajamento. Se uma estação E_s apresentar, para uma das filas F_i, um ponteiro $P_{Fi}(j)$ tal que:

$$P_{Fi}(j) < P_{EGi}(j),$$

então, a proposição de mensagens para engajamento através do ponteiro $P_{EGi}(j)$ pela $E_c(j)$, não está completamente presente nas filas F_i da estação E_s. As mensagens não presentes serão recuperadas então, através de um pedido-de-retransmissão à estação emissora das mensagens perdidas.

Verificadas e satisfeitas as duas condições C1 e C2, a estação E_s envia a $E_c(j)$ um reconhecimento positivo ao pedido-de-commit(j). Se C1 não for satisfeita E_s deve enviar um reconhecimento negativo ao pedido-de-commit(j). Se C1 for satisfeita mas não C2, E_s deve primeiro recuperar a mensagem ou commit perdido, para depois, se ainda for possível, reconhecer positivamente ao pedido-de-commit(j).

Tratamento das mensagens de reconhecimento por $E_c(j)$

Uma vez difundido o pedido-de-commit(j), $E_c(j)$ fica aguardando o reconhecimento das outras estações. Duas situações podem ocorrer então:

- **$E_c(j)$ recebe um ou mais reconhecimentos negativos**: neste caso é necessário abortar a execução de protocolo j pois $E_c(j)$, tentou impor uma lista $L_{mc}(j)$, que não reflete o estado de engajamento das F_i das estações $E_i \subset G_L(j)$ (condição C1 violada: existem ponteiros não apropriados).

- **Expira o tempo de espera** (T_E) em $E_c(j)$ e os **reconhecimentos** recebidos são todos **"positivos"**: neste caso tem-se ainda duas possibilidades. Na primeira a minoria das estações reconheceu positivamente ao "pedido-de-commit". Nesta situação, diz-se que não foi formado

um "commit quorum" e portanto, $E_c(j)$ deve abortar a execução de engajamento j (situação de fracasso), reiniciando-a após um timeout. Na outra situação (situação de sucesso), a maioria das estações reconheceu positivamente ao pedido-de-commit(j). Houve então a formação de um "commit quorum". A estação $E_c(j)$ pode enviar a mensagem commit(j), que determina o engajamento das mensagens, apontadas por $L_{me}(j)$, em todas as estações. A mensagem commit(j) transfere também o token para a nova coordenadora ($E_c(j+1)$).

Mensagem de "commit"

Tendo recebido o reconhecimento positivo da maioria das estações participantes, a mensagem commit é difundida, devendo conter um campo para cada um dos seguintes itens:

- **Coordenadora**: $E_c(j)$, isto é, estação emissora da mensagem commit(j) e portadora do token.
- **Nova coordenadora**: $E_c(j+1)$, isto é, estação a quem o token será transferido.
- **Versão-token**: número da configuração atual do anel.
- **N-commit** : número do atual commit.
- $\mathbf{L_{me}(j)}$: isto é, ponteiro da última mensagem a engajar de cada uma das estações $E_i \in G_L(j)$.
- **Grupo-R**: contém o nome das R estações que irão armazenar as mensagens que serão engajadas neste commit. As R estações, além da coordenadora ($E_c(j)$) escolhidas são as R primeiras estações a reconhecer o pedido-de-commit(j). Estas (R+1) estações serão as responsáveis pela restauração do commit(j) para estações que venham a solicitá-lo.
- **Grupo-Ausentes**: $G_A(j)$, isto é, contém nome das estações que não reconheceram ao pedido-de-commit(j).
- **Grupo-falho**: $G_f(j)$, isto é, contém nome das estações que não participaram das M últimas execuções de protocolo.
- **Grupo-Recuperando**: $G_r(j)$ contém nome das estações que enviaram pedido-de-recuperação-de-commit(j)-perdido mas ainda não o reconheceram. Encontrando-se em processo de recuperação do commit(j).

Afim de garantir que, seja qual for a estação nova coordenadora ($E_c(j+1)$), esta estará habilitada a enviar as mensagens "commit(j)" (ou ainda commits anteriores) às estações que estão em processo de recuperação de commit, todas estações ao receberem a mensagem "commit(j)" colocam-na em uma Fila-M-msg-commit, ordenada pelo número do commit e com capacidade M. A mensagem de commit(j) é fundamental para uma estação em recuperação poder localizar as estações que mantêm ainda cópias das mensagens engajadas na execução de protocolo j (o Grupo-R do engajamento).

As estações fazem também o engajamento das mensagens especificadas por $E_c(j)$ na ordem do campo $L_{me}(j)$. Caso a estação pertença ao grupo-R ,esta deverá ainda armazenar as L mensagens a engajar na **"Lista-Regeneração"**. Estas mensagens serão mantidas armazenadas por M execuções de protocolo, quando então serão descartadas. As estações, ao receberem a mensagem "commit(j)", devem ainda deletar de $G_p(j)$ as estações pertencentes ao Grupo-falho $G_f(j)$.

Passagem do Token

A nova coordenadora ($E_c(j+1)$) será, a princípio a estação possuidora da última bufferização $B_i(j)$ que completará $B_L(j)$. Se no entanto, esta estação (a pretendida $E_c(j+1)$) não tiver respondido ao pedido-de-commit(j), a estação E_i ($E_i \in G_L(j)$) imediatamente anterior será tomada, então, como a nova coordenadora, desde que, tenha respondido ao pedido-de-commit(j). Neste caso, a pretendida $E_c(j+1)$ é retirada de $G_L(j)$ e suas mensagens de $B_L(j)$. O protocolo não admite que seja transferido o token a uma estação ausente. As únicas mensagens, de estações ausentes, que serão desconsideradas pela mensagem "commit(j)" no engajamento j (na bufferização $B_L(j)$) serão das estações que, com a nova proposição de $E_c(j+1)$, ficarem fora do $G_L(j)$, delimitado apartir desta nova coordenadora .

O protocolo também não admite que $E_c(j)$ transfira o token para si mesma. Caso as condições favoreçam esta decisão, a execução do protocolo deve ser abortada e reiniciada após um timeout.

AXIOMA 1: A condição para a não ruptura do anel lógico é que:

$$E_c(j),\ E_c(j+1) \in \{G_L(j) \cup G_{EG}(j)\} \quad \wedge \quad E_c(j) \neq E_c(j+1) \tag{3.5}$$

3.3.2.4 - Tratamento de excessões e robustez do algoritmo

Como é permitido a uma estação se ausentar de até M execuções de protocolo sem ser considerada faltosa (grau de omissão M) é necessário criar um suporte para recuperação desta estação. Assim, as mensagens engajadas em cada execução de protocolo m, serão mantidas armazenadas por R estações além da $E_c(m)$ durante M execuções de protocolo. Com isto, mesmo que R estações venham a faltar, uma estação que tenha se ausentado de menos de M execuções de protocolo pode ainda vir a recuperar as mensagens e continuar operando sem a necessidade de reinserção ao sistema (sistema R-resistente).

Afim de determinar se uma estação está em estado de "crash" ou se está ausente por omissão, foram criadas **listas-de-observação**, contendo os M-1 Grupos-Ausentes, $\{G_A(j-M+1), G_A(j-M+2), \ldots, G_A(j-1)\}$, um para cada um dos (M-1) últimos commits executados.

Uma estação E_i pode detectar a perda de um commit(m) em uma execução de protocolo de ordem, no máximo, $j = m+M-1$. Diante desta situação, E_i envia à $E_c(j)$ um pedido-de-recuperação-de-commit(m)-perdido. Com isto E_i inicia a execução de recuperação da execução de engajamento m sendo colocada no $G_r(m)$. Ao recuperar o commit da execução de protocolo m, E_i é

retirada de $G_r(m)$ e $G_A(m)$. Os pedidos de recuperação serão sucessivos até que a estação atinja a execução de protocolo atual.

Uma estação E_i se encontrando nos (M-1) Grupos-Ausentes (G_A) da Lista-de-observância e estando fora dos últimos (M-1) Grupos-Recuperando (G_r) será considerada faltosa em $G_f(j)$ e retirada do $G_p(j+1)$:

$$G_f(j) = \left\{ E_i \mid E_i \in \bigcap_{m=j-M+1}^{m=j} G_A(m) \quad \wedge \quad E_i \notin \bigcup_{m=j-M+1}^{m=j} G_r(m) \right\}$$

No momento em que $G_A(m) = \varnothing$ tem-se a situação em que todas as estações ou recuperam o commit(m) ou foram retiradas do sistema, não havendo necessidade de continuar a armazenar as mensagens engajadas nas (M-1) execuções de protocolo anteriores a m. Portanto as R estações, que mantêm cópias destas mensagens na camada de difusão deverão descartá-las na situação de $G_A(m)$ sem elementos. Outra situação de descarte de mensagens pelo grupo-R de um commit é quando a distância deste em relação a execução de protocolo atual for maior que M.

Perda de commit

Uma estação E_i percebe que perdeu um commit em três situações:

- **Recebeu a mensagem "commit(j)" sem ter recebido e/ou reconhecido o pedido-de-commit(j)**: estando em dia com os commits anteriores e estando habilitado a engajar as mensagens propostas por $E_c(j)$, E_i faz o commit normalmente. No caso de não possuir todas mensagens que serão engajadas, E_i deve recuperá-las e, só então engajar as mensagens. As informações para recuperar mensagens a engajar são conseguidas na mensagem commit(j) (item anterior).

- **Recebeu um pedido-de-commit(j) e não recebeu mensagem de commit correspondente**: E_i difunde a mensagem pedido-de-recuperação-de-commit(j)-perdido para tentar receber a decisão. Se commit(j) não é recebida mesmo assim, E_i considera que $E_c(j)$ está faltosa e tenta restaurar o token.

- **Recebeu um pedido-de-commit(j) de número não sequencial ao último commit em que participou**: sendo a diferença entre o último commit executado por E_i e o pedido-de-commit(j) menor ou igual a M e maior que um (1) E_i perdeu algum(ns) commit(s) mas ainda não foi considerada faltosa. Assim, E_i deve enviar à $E_c(j)$ um "pedido-de-recuperação-de-commit-perdido" contendo o número do primeiro commit perdido.

Uma vez recebido o pedido de recuperação de um commit(r) perdido, a coordenadora coloca a estação requisitante no $G_r(r)$ e lhe envia a mensagem de commit(r). A estação então, recupera com uma das estações do grupo-R cada uma das mensagens a engajar perdidas, se houver, e executa o commit

perdido. Ao completar o commit(r) esta difunde uma mensagem de reconhecimento ao pedido-de-commit(r) ao sistema. Com isto as estações pertencentes a $G_p(j)$ retiram E_i de $G_A(r)$ e de $G_r(r)$. Isto é repetido sucessivamente até a recuperação de todos commits perdidos pela estação requisitante.

Perda de mensagens

A perda de mensagens pode ser detectada em três situações, e apesar de uma mesma mensagem de pedido-de-retransmissão ser enviada, cada situação tem o seu "receptor de pedido" apropriado. Um "pedido-de-retransmissão" deve conter três campos:

1. Estação E_i que difundiu a mensagem $m_{i,k}$
2. Ponteiro da primeira mensagem $m_{i,k}$ perdida
3. Estação E_s que difundiu pedido de retransmissão

As três situações de detecção de perda de mensagens correspondem a:

- **Detecção da perda na recepção de uma nova mensagem difundida pelo mesmo emissor**: nesta situação o pedido-de-retransmissão deve ser enviado ao emissor correspondente.
- **Detecção da perda na verificação de um pedido-de-commit**: não constando em suas filas F_i todas mensagens que deverão ser engajadas, segundo a coordenadora, a estação que as perdeu deve requisita-las ao emissor.
- **Detecção da perda na recuperação de commit perdido**: nesta situação, após a estação receber a mensagem de commit perdida e detectar a perda de mensagens de uma fila F_i qualquer, engajadas neste commit, esta envia um pedido-de-retransmissão a uma estação do grupo-R responsável pelo armazenamento destas mensagens após seus engajamentos nas estações presentes no grupo das engajantes (G_{EG}) do commit.

Detecção de estações faltosas

Toda estação E_i que não reconhece a um pedido-de-commit(m) é colocada no $G_A(m)$. Se:

$$E_i \in \bigcap_{m=j-M+1}^{m=j} G_A(m) \quad \wedge \quad E_i \notin \bigcup_{m=j-M+1}^{m=j} G_r(m)$$

E_i será considerada em "crash" e inserida em $G_f(j)$. Será retirada de $G_p(j+1)$, fazendo parte então de $G_F(j+1)$.

Reconfiguração do token

Após a estação E_i participar do grupo de estações-L de um engajamento (m) ($E_i \in G_L(m)$), é acionado um timeout, devendo a estação então participar novamente do grupo de estações-L antes deste tempo expirar. O timeout, denominado de T_{reconf}, deverá cobrir o anel "máximo", ou seja, na pior situação a estação E_i voltará a ser estação-L em NK/L execuções de protocolo. Assim:

$$T_{reconf} > \left(\frac{NK}{L} \times (\textit{tempo máximo de uma execução do protocolo}) \right)$$

A figura 4.1 ilustra as relações de tempo envolvidas na execução de um protocolo e o item 4.4 explica a faixa de valores deste tempo.

Se, após T_{reconf}, E_i não for novamente estação-L, E_i considera que o token foi perdido ($E_c(j)$ em crash), invocando então sua regeneração às demais estações pertencentes a $G_p(j)$. Para investigar a existência do token na rede e regenerar de forma única e correta o token, é usado o algoritmo de regeneração proposto por [Nishio 91]. Neste algoritmo são definidas as seguintes estruturas, necessárias:

- **Versão-estação$_i$**: versão de E_i, um inteiro não negativo armazenado em cada estação E_i, tal que $i = [1, 2, ..., N]$.
- **Versão-token**: versão do token, um inteiro não negativo armazenado no token.

Tanto Versão-estação$_i$ quanto Versão-token são estabelecidas inicialmente com valor zero. Quando uma estação E_i, denominada de Estação Regeneradora suspeita da perda de token, esta executa o Algorítmo de Regeneração, estabelecendo um valor de **Versão-proposta$_i$** para o "versão-token". E_i, então, atualiza a versão-estação$_i$ com o valor da versão-proposta$_i$ e envia uma mensagem Token-perdido$_i$ perguntando a cada estação E_s ($s \neq i$) se o token com a versão-proposta$_i$ pode ser regenerado pela E_i. Cada E_s ao receber o Token-perdido$_i$ executa um procedimento de Resposta que compara a "versão-estação$_s$" com a "versão-proposta$_i$". Se versão-proposta$_i$ for maior que versão-estação$_s$ e E_s não for possuidora do token, a versão-estação$_s$ é atualizado com o valor da versão-proposta$_i$ e um reconhecimento positivo é enviado a E_i. Em caso contrário, um reconhecimento negativo é enviado a E_i. O reconhecimento negativo é necessário para garantir que somente a estação geradora da maior versão-proposta$_i$ regenere o token, e que, no caso de uma estação já possuir o token, esta evite a regeneração. O algoritmo completo de regeneração do token é apresentado no apêndice A.

3.4 - Conclusão

Este capítulo apresentou uma proposta de protocolo de difusão confiável, a qual incorpora a semântica de faltas de omissão, com grau de omissão M. O protocolo possui grau de "recuperação" ("resiliency") R permitindo que, mesmo na presença de R estações faltosas, uma estação que tenha se omitido por menos de M execuções consecutivas possa se recuperar. Tal proposta apresentou um engajamento de L mensagens por execução sendo a ordenação destas mensagens dada em função da ordem local das mensagens e sua localização no anel virtual.

No próximo capítulo os diversos parâmetros definidos na apresentação desta proposta serão analisados no sentido de limitar a escolha de valores de projeto para estes.

CAPÍTULO 4

DISCUSSÃO SOBRE O DESEMPENHO E PARÂMETROS DO ALGORITMO

4.1 - Introdução

Como foi apresentado em 3.2.1, os parâmetros para a escolha e/ou elaboração de um algoritmo estão relacionados com aspectos de desempenho. Neste sentido será dada neste capítulo uma visão mais ampla dos parâmetros introduzidos e suas relações, de modo a se ter um bom desempenho na execução do algoritmo, minimizando o overhead (sobrecarga) e a latência na difusão confiável de cada mensagem.

4.2 - Análise dos parâmetros

Dos quatro parâmetros apresentados (K, L, M, R), M e R são dependentes apenas dos recursos de memorização das estações envolvidas, de forma que, não determinam nenhuma sobregarga (overhead) adicional ao algoritmo. O parâmetro R indica o número de estações que manterão, na camada de difusão confiável, as mensagens engajadas em uma execução de protocolo. Este parâmetro determina a robustez do algoritmo no sentido de se poder restaurar engajamentos executados em estações que estiveram ausentes. Esta possibilidade de restauração é garantida mesmo na presença de R "crash" de estações. O parâmetro M determina o número de execuções consecutivas de protocolo que uma estação pode se ausentar sem ser tida como faltosa, isto é, M determina o grau de omissão do sistema. Este parâmetro M está envolvido apenas com listas como a Lista-de-observância e Fila-M-msg-commit, que produzem tão somente custos com recursos de memória.

Os parâmetros K e L estão relacionados com as filas F_i e F_L respectivamente e, consequentemente com as bufferizações $B_i(j)$ e $B_L(j)$. O desempenho do protocolo depende destes valores e suas relações. Pelas condições de circulação do token e baseado na relação (3.5) em que a coordenadora $E_c(j)$ não pode continuar com o token na execução de protocolo (j+1), tem-se que:

$$(K + 1) \leq L \leq NK \qquad (4.1)$$

o limite inferior garante que não serão engajadas somente mensagens de $E_c(j)$. O limite superior garante que L é inferior ou igual a capacidade de armazenamento do anel (NK), e logo, não haverá a circulação completa do anel em uma passagem do token.

4.3 - Sobrecarga de mensagens de controle

Sob condições normais de execução de protocolo, com formação de "commit quorun", em uma rede de topologia física qualquer, envolvendo N estações participantes e ligações ponto-a-ponto (que é o pior caso) o número máximo de mensagens-protocolo inclue:

- N mensagens "pedido-de-commit"
- N mensagens de reconhecimento positivo do pedido-de-commit
- N mensagens "commit"

o número total de mensagens trocadas nestas condições será igual a 3N mensagens-protocolo, isto é, 3N mensagens-protocolo trafegam pela rede por execução de protocolo.

O overhead então, é dado pelo número de mensagens-protocolo que trafegam pela rede, por mensagem engajada (secção 3.2.1) . Ou seja,

$$overhead = \frac{3N}{L}$$

Logo, quanto maior for a capacidade de engajamento por execução, menor o custo de overhead provocado pelo protocolo. Para L >> N o overhead é nulo. Em relação a K, com base em (4.1) que define os limites de variação de L, tem-se:

$$\frac{3N}{NK} \geq overhead \geq \frac{3N}{K+1} \Rightarrow \frac{3}{K} \geq overhead \geq \frac{3N}{K+1}$$

Assim, o overhead é inversamente proporcional ao valor de K.

4.4 - Latência

Latência de uma mensagem é definida como o tempo em que esta mensagem pode permanecer na camada de difusão confiável, antes de seu engajamento. Para a determinação destes valores são introduzidas as definições de alguns tempos relacionados com a execução do protocolo:

t_{Aj} : é o tempo onde inicia, em uma execução de protocolo j, a ativação do processo de engajamento das mensagens propostas nesta execução de protocolo.

t_{EGj}: é o tempo de duração da execução do engajamento j.

t_{EPj}: é o tempo de duração da execução de protocolo j.

A figura 4.1 mostra estes tempos, que tem como origem o inicio da execução de protocolo. Da figura obtem-se a relação:

$$t_{EPj} = t_{Aj} + t_{EGj}$$

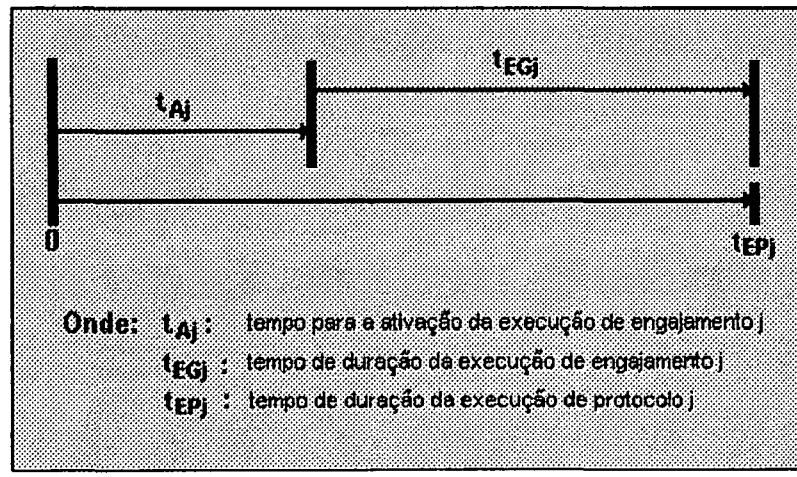


Figura 4.1 - Tempo de uma execução de protocolo

No protocolo, é assumido as seguintes condições limites em relação a estes tempos:

T_A :é o "timeout" para a ativação de uma execução de engajamento.

$$0 \leq t_{Aj} \leq T_A \tag{4.2}$$

T_E :é o "timeout" limitando a espera de respostas ao pedido-de-commit

$$T_E < t_{EGj} \tag{4.3}$$

Com base nestas definições podemos estabelecer certas condições para uma mensagem $m_{i,k}$ que chega na camada de difusão confiável. Esta mensagem terá duas situações possíveis de latência: ser engajada na execução de protocolo de sua chegada na camada ou ainda, em duas ou mais execuções de protocolo sucessivas, apartir de sua chegada na camada de difusão. Destas situações, a priori não se pode concluir o caso extremo de latência.

No sentido de facilitar o exame das possibilidades dois cenários são introduzidos:

Cenário 1

Neste cenário é considerado que a mensagem $m_{i,k}$ chega na execução de protocolo j e é engajada nesta mesma execução de protocolo. A figura 4.2, ocasião 1, esboça este cenário. A chegada da mensagem no intervalo t_{EGj} (ocasião 2, figura 4.2) implica que a mesma não poderá ser engajada nesta execução de protocolo (j), caindo fora do cenário 1.

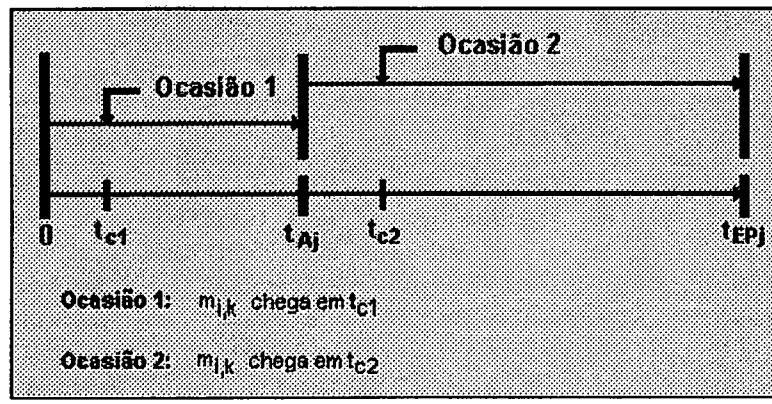


Figura 4.2 - Ocasião 1 e 2 de chegada de mensagem

O engajamento da mensagem na execução de protocolo j implica então, na chegada com tempos inferiores a t_{Aj}. Considerando t_{c1}, o tempo de chegada da mensagem $m_{i,k}$ (a origem deste tempo também

é no início da execução de protocolo), para o engajamento de $m_{i,k}$ na execução de protocolo j, tem-se então:

$$0 \leq t_{cl} \leq t_{Aj}$$

Com isto, a latência de mensagem é dada por:

$$Lat = t_{Aj} - t_{cl} + t_{EGj}$$

Como o interesse é caracterizar a pior situação, é assumido neste cenário $t_{cl} = 0$ ou seja, a mensagem chega na origem da execução de protocolo j. Nesta situação a latência resume-se a:

$$Lat = t_{Aj} + t_{EGj} \quad (4.4)$$

A equação (4.4) é assumida para o engajamento sendo ativado pelo preenchimento de L mensagens na camada de difusão confiável ($B_L(j) = L$). Este cenário poderia, no entanto, ser decomposto em outra situação onde $B_L(j) \leq L$, .neste caso ter-se-ia $t_{Aj} = T_A$ para $B_L(j) < L$, determinando uma latência máxima para este cenário igual a:

$$Latmax = T_A + t_{EGj} \quad (4.5)$$

Cenário 2

Neste cenário a mensagem $m_{i,k}$ chega após o inicio da execução de engajamento j-1 e antes do inicio da execução de protocolo j (figura 4.3, ocasião 3), não podendo mais ser engajada em j-1. Esta mensagem portanto deverá ser engajada em uma ou mais execuções de protocolo subsequentes a execução de protocolo j-1.

Como a recepção de mensagens difundidas sobre a rede é contínua, a latência de $m_{i,k}$ irá depender do número de mensagens que serão recebidas, (armazenadas no anel virtual) e que antecedem $m_{i,k}$ segundo a ordem "$\Rightarrow$". Estas mensagens chegam desde a execução de protocolo j-1 até a execução onde $m_{i,k}$ é engajada e deverão posicionar-se, no anel virtual, entre $E_c(j)$ e a mensagem $m_{i,k}$. Tais mensagens serão consideradas como sendo a bufferização $B(m_{i,k})$ e serão engajadas apartir da execução de protocolo j.

Para determinar a latência de $m_{i,k}$, é assumido que esta será engajada em E execuções de protocolo, contando apartir da execução j :

$$0 \leq (E - 1)L \leq card(B(m_{i,k})) \leq EL$$

com E inteiro positivo.

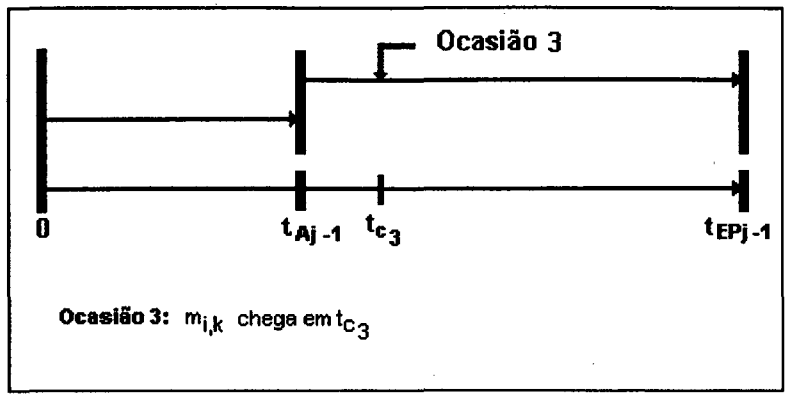


Figura 4.3 - Ocasião 3 de chegada de mensagem

O pior caso de latência neste cenário, e que será mantida neste texto, corresponde ao da mensagem com tempo de chegada coincidindo com o início da ativação do engajamento j-1 ($t_{c3} = t_{Aj-1}$). Neste caso, o tempo de permanência de $m_{i,k}$ na camada de difusão corresponde ao período após a chegada na execução de protocolo j-1, completando E+1 execuções de protocolo:

$$Lat = t_{EGj-1} + \sum_{m=j}^{m=j-1+E} (t_{Am} + t_{EGm})$$

Assumindo card(B($m_{i,k}$)) sendo igual a EL, ou seja, a mensagem $m_{i,k}$ sendo a EL^{ésima} mensagem na ordenação para o engajamento (apartir da execução j), tem-se que o inicio de cada execução do protocolo subsequente ao engajamento j -1 possuirá t_{Am}= 0. Isto porque as condições de engajamento (L mensagens) são satisfeitas na origem de cada execução de protocolo subsequente a j-1. Resumindo-se então a latência a:

$$Lat = \sum_{m=j-1}^{m=j-1+E} t_{EGm} \qquad (4.6)$$

A expressão (4.6) vale para situações onde B($m_{i,k}$) é um múltiplo de L. Assumindo uma situação contrária, onde:

$$0 \leq (E-1)L < B(m_{i,k}) < EL$$

a latência toma então a sequinte forma:

$$Lat = t_{A_{j-1+E}} + \sum_{m=j-1}^{m=j-1+E} t_{EG_m}$$

para esta expressão, a pior situação de latência será quando $t_{Aj-1+E} = T_A$:

$$Latmax = T_A + \sum_{m=j-1}^{m=j-1+E} t_{EGm} \qquad (4.7)$$

Em resumo, as latências máximas da mensagem $m_{i,k}$, nos cenários descritos são então:

Cenário1: $$Latmax = T_A + t_{EGj} \quad (4.5)$$

Cenário2: $$Latmax = \sum_{m=j-1}^{m=j-1+E} t_{EGm} \quad ,para \quad EL = B(m_{i,k}) \quad (4.6)$$

$$Latmax = T_A + \sum_{m=j-1}^{m=j-1+E} t_{EGm} \quad ,para \quad (E-1)L<B(m_{i,k})<EL \quad (4.7)$$

4.4.1 - Efeitos dos parâmetros de projeto K e L sobre o desempenho do algoritmo

Nos cenários apresentados as expressões (4.5), (4.6) e (4.7) não permitem concluir sobre os parâmetros K e L e muito menos, examinar o desempenho do algoritmo em diferentes situações de carga. Para que se possa estabelecer algumas considerações, assume-se a execução do protocolo com N estações participantes e o valor δ, como o tempo de transmissão de uma mensagem no suporte de comunicação. A topologia física da rede é assumida qualquer, envolvendo ligações ponto a ponto[1] . Neste caso:

$$t_{EGj} \leq 3N\delta \quad (4.8)$$

onde 3N representa o número de mensagens-protocolo trocadas no engajamento de uma execução de protocolo.

Considerando a situação sem exceção, com as N estações respondendo ao pedido-de-commit(j) no periodo limitado por T_E pode-se dimensionar T_E em:

$$T_E = 2N\delta \quad (4.9)$$

onde 2N corresponde às mensagens de pedido-de-commit(j) e as respectivas respostas, emitidas dentro do limite T_E. Com base nas relações (4.3), (4.8) e (4.9) e na condição das N estações participantes, assumiu-se então:

$$t_{EGj} = T_E + N\delta$$

[1] Neste tipo de topologia, δ poderia ser assumido como o valor médio dos tempos de transferência de mensagens entre dois pontos quaisquer da rede. No presente caso, o interesse são os limites máximos e sabendo ainda, que a circulação do token não garante a participação de todas as estações como coordenadoras na execução do protocolo, descaracterizando portanto a idéia de tempo médio. Então com base nisto, é assumido que δ é o valor de tempo máximo na transferência de mensagens no suporte de comunicação.

onde Nδ representa a emissão de N mensagens de commit. Como o segundo membro é uma constante, introduz-se então, T_{EG} como o tempo limite de engajamento:

$$t_{EGj} = T_{EG} = T_E + N\delta = 3N\delta \qquad (4.10)$$

Em relação ao dimensionamento de T_A (timeout para a ativação do engajamento), é levado em conta as necessidades de tempo para a ocupação da capacidade de engajamento (L mensagens) na camada de difusão confiável. Diante disto, é assumido:

$$T_A = \sigma L\delta$$

onde σ é uma constante, com $\sigma > 1$

Com os valores assumidos para T_A e t_{EGj}, os cenários descritos anteriormente são retomados:

Cenário 1

Retornando à expressão (4.5) da latência, com os valores de T_A e t_{EGj}, tem-se:

$$\text{Latmax} = \sigma L\delta + 3N\delta$$

No cenário 1, a mensagem $m_{i,k}$ é engajada em uma execução de protocolo. Para examinar a dependência do desempenho do protocolo em função dos valores de L, na expressão da latência máxima são testados valores extremos de L:

para L = NK $\quad$ Latmax = σ (NK) δ + 3Nδ

rearranjando a expressão da latência em função da constante T_{EG}, tem-se:

$$Latmax = \frac{\sigma}{3} KT_{EG} + T_{EG}$$

para L = K+1 $\quad$ Latmax = σ(K+1)δ + 3Nδ

rearranjando:

$$Latmax = \frac{\sigma(K+1)}{3N} T_{EG} + T_{EG}$$

Assumindo $\sigma = \frac{3N}{(K+1)}$ com $N > \frac{(K+1)}{3}$ tem-se então:

Para L = NK

$$Latmax = N \binom{K}{K+1} T_{EG} + T_{EG} \qquad (4.11)$$

Para L=K+1

$$Latmax = 2T_{EG} \qquad (4.12)$$

Nas equações (4.11) e (4.12) é explicitado que o engajamento de uma mensagem no periodo de uma execução de protocolo favorece as situações de L assumindo valores pequenos (L = K+1). Isto vale no cenário 1, tanto para situações de ocupação da capacidade de engajamento ($B(m_{i,k}) = L$) como para situações de subutilização ($B(m_{i,k}) < L$).

Cenário 2

Retomando as equações (4.6) e (4.7) com os valores de T_A e t_{EGj}, tem-se:

Para $B(m_{i,k}) = EL$

$$Latmax = \sum_{m=j-1}^{m=j-1+E} t_{EGm}$$

$$Latmax = (E + 1)\, T_{EG} \qquad (4.13)$$

Para $(E-1)L < B(m_{i,k}) < EL$

$$Latmax = T_A + \sum_{m=j-1}^{m=j-1+E} T_{EGm}$$

$$Latmax = T_A + (E + 1)\, T_{EG} \qquad (4.14)$$

O número máximo de mensagens que podem preceder a mensagem $m_{i,k}$ é de (NK-K+L-1) mensagens. Nesta situação de carga, card($B(m_{i,k})$) é igual a (NK- K) e a mensagem $m_{i,k}$ é a (NK-K)^{ésima} mensagem pela ordenação "$\Rightarrow$" na bufferização $B(m_{i,k})$. A figura 4.4 ilustra um exemplo deste tipo de situação, onde a mensagem $m_{i,k}$ chega em t_{Aj-1}, com o engajamento já iniciando (figura 4.4a). Neste exemplo a mensagem $m_{i,k}$ será engajada em E+1 = 4 execuções de protocolo, com L = 4 e $m_{i,k}$ como a décima-segunda mensagem de $B(m_{i,k})$ (décima-sexta mensagem contando apartir de $E_c(j-1)$). Em vista disto e retomando as expressões (4.13) e (4.14), tem-se:

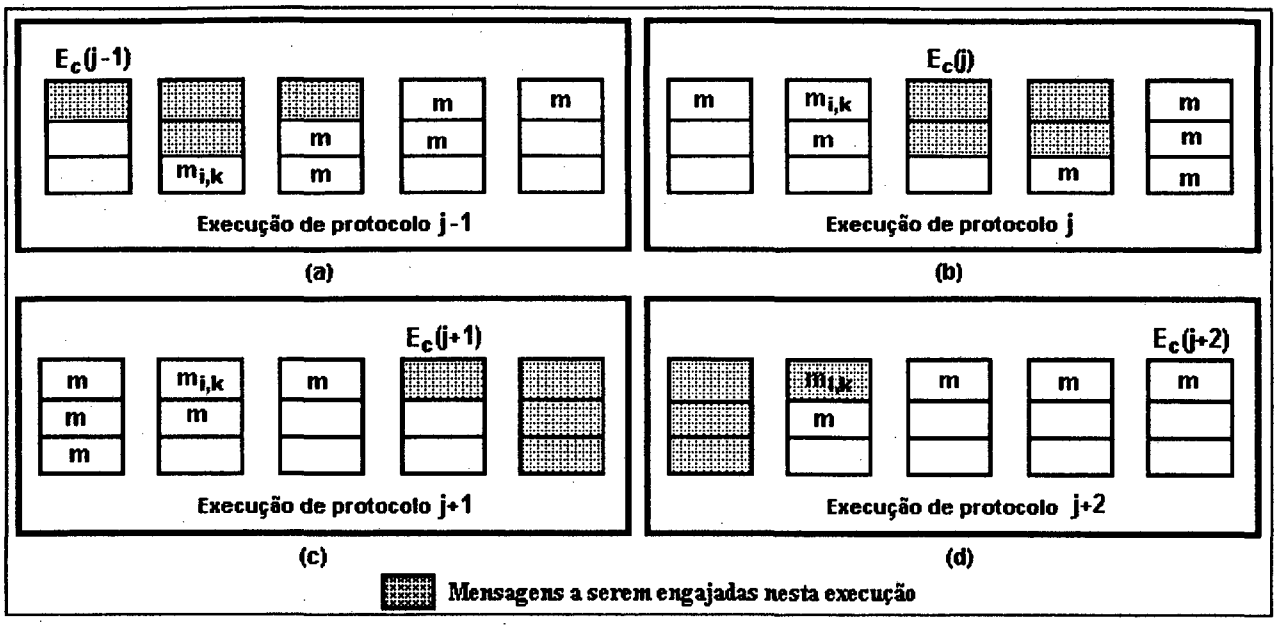


Figura 4.4 - Exemplo da pior situação para o cenário 2 para N=5, K=3 e L=4

Para L=K+1 e com EL=NK-K

$$Latmax = \left(\frac{NK - K}{K + 1} + 1\right) T_{EG} = \left(\frac{NK + 1}{K + 1}\right) T_{EG}$$

Rearranjando a expressão:

$$Latmax = (N - 1)\left(\frac{K}{K + 1}\right) T_{EG} + T_{EG}$$

Para L = K+1, (E-1)L < NK-K < EL e σ = 3N/ (K+1)

$$Latmax = \frac{3N}{K + 1}(K + 1)\delta + \left(\left\lceil\frac{NK - K}{K + 1}\right\rceil + 1\right) T_{EG} = T_{EG} + \left(\left\lceil\frac{NK + 1}{K + 1}\right\rceil\right) T_{EG}$$

Rearranjando a expressão:

$$Latmax = \left(\left\lceil(N - 1)\left(\frac{K}{K + 1}\right)\right\rceil\right) T_{EG} + 2T_{EG} \qquad (4.16)$$

Em situações de L igual a NK, para que se tenha o engajamento de $m_{i,k}$ em mais de uma execução de protocolo, com esta chegando em t_{Aj-1}, é necessário que o engajamento j-1seja ativado por esgotamento de timeout ($t_{Aj-1} = T_A$). Só deste modo mensagens que não serão engajadas em j-1 poderão ser armazenadas na execução de protocolo j-1. Nesta situação ($B_L(j-1) < L$), $m_{i,k}$ chegando em t_{Aj-1} e o engajamento (j) sendo ativado com L = NK em $t_{Aj} = T_A$:

$$Latmax = N\left(\frac{K}{K+1}\right)T_{EG} + 2T_{EG} \qquad (4.17)$$

As equações de latência para os diferentes cenários podem então ser resumidas na tabela 4.1 considerando

$$\sigma = \frac{3N}{K+1} \quad e \quad N \geq \frac{K+1}{3}$$

<table>
<tr><th></th><th>L = NK</th><th colspan="2">L = K + 1</th></tr>
<tr><td>Cenário 1</td><td>$Latmax = N\left(\frac{K}{K+1}\right)T_{EG} + T_{EG}$</td><td colspan="2">2 T_{EG}</td></tr>
<tr><td rowspan="2">Cenário 2</td><td rowspan="2">$Latmax = N\left(\frac{K}{K+1}\right)T_{EG} + 2T_{EG}$</td><td>EL = NK - K</td><td>0≤(E-1)L<NK-K<EL</td></tr>
<tr><td>$Latmax = (N-1)\left(\frac{K}{K+1}\right)T_{EG} + T_{EG}$</td><td>$Latmax = \left(\left[(N-1)\left(\frac{K}{K+1}\right)\right]\right)T_{EG} + 2T_{EG}$</td></tr>
</table>

Tabela 4.1 - Latência Máxima

4.4.2 - Comentários

Das equações acima (tabela 4.1) é evidenciado que mesmo em situações onde o suporte de comunicação é sobrecarregado, ($B(m_{i,k})$ = NK-K), a opção de L pequeno (L = K+1) não representa uma situação de latência pior que as de L = NK. Embora, se deva considerar que em termos de overhead, o custo de mensagens de controle por mensagem engajada é maior para L = K+1. Em termos de subutilização ou da simples ocupação da capacidade de engajamento (L), o desempenho do algoritmo para L = K+1 é muito superior se comparado com valores de L próximos do máximo (cenário 1). A tabela 4.2 e a figura 4.5 servem para uma melhor comparação quantitativa das distintas opções de escolha dos

parâmetros de projeto, considerando as diferentes situações de carga do suporte de difusão confiável, em uma rede com N=12 estações.

K	Overhead		Latência [T_{EO}]			
			Cenário 1		Cenário 2	
	L_{min}	$L_{máx}$	L_{min} = K+1	L_{max} = NK	L_{min} = K+1	L_{min} = NK
1	18	3	2	7	8	8
3	9	1	2	10	11	11
5	6	0,6	2	11	12	12
8	4	0,38	2	11,667	12	12,667
11	3	0,27	2	12	13	13
12	2,77	0,25	2	12,077	13	13,077
19	1,8	0,16	2	12,4	13	13,4
23	1,5	0,13	2	12,5	13	13,5
28	1,24	0,11	2	12,586	13	13,586
31	1,13	0,1	2	12,625	13	13,625
34	1,03	0,09	2	12,657	13	13,657

Tabela 4.2 - Valores de overhead e latência no espectro K

Um aspecto que pode ser levantado da tabela 4.2 e figura 4.5 é que com as várias opções de cenários e de valores de L, e $\sigma = 3N/(K+1)$, as latências tendem a valores limites ($2T_{EG}$ e $(N+2)T_{EG}$) com o aumento de K. Estes valores limites são suficientes, em N=12, para darem vazão no sentido da ocupação da capacidade de engajamento (L) e permitirem a execução do protocolo, até o engajamento das mensagens consideradas no cálculo da latência.

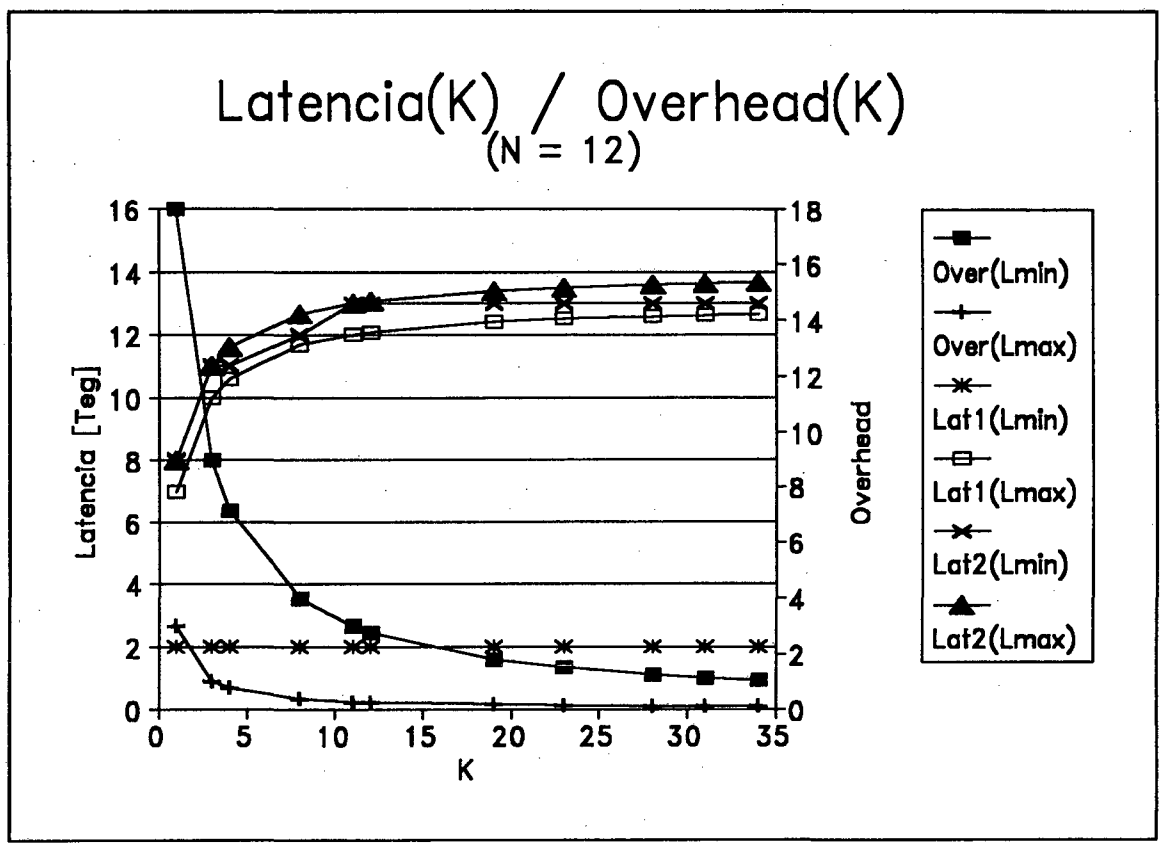


Figura 4.5 - Latência [T_{EG}]/Overhead x K

Se como suporte de comunicação for considerada uma rede Ethernet (CSMA/CD) com topologia física multiponto, as trocas necessárias para o engajamento de L mensagens correspondem a N+2 mensagens-protocolo. Com isto, T_{EG} assume o valor $(N+2)\delta$. Tomando $\sigma = (N+2)/(K+1)$, as expressões obtidas para a rede Ethernet, para os mesmos cenários, são idênticas as da tabela 4.1.

Uma rede CSMA/CD, como apresenta acesso probabilista, não permite o uso de δ_{max}. Isto determina o uso de um valor médio para δ . Assim, com 12 estações emitindo continuamente na taxa de 10 Mbps e mensagens de 64 bytes, o δ médio é assumido como 2 [ms/mensagem] [Kümmerle 87].

Assumindo $\delta = 2$ [ms/mensagem], tem-se $T_{EG} = 28$ mseg, e com isto:

Lat max = 56 [ms], para L = K+1, cenário 1

Lat max = 364 [ms], para L = NK, cenário 1 e L = K+1, cenário 2

Lat max = 392 [ms], para L = NK, cenário 2

Estes números dão a idéia do desempenho do protocolo em uma rede local Ethernet, em condições limites de ocupação do meio de comunicação.

4.5 - Comentários sobre o protocolo proposto

Este trabalho apresenta um algoritmo que, como [Chang 84], [Melliar-Smith 90] e outros se enquadra na classe de protocolos de difusão atômica assíncrona. No entanto, seu tempo máximo por execução de protocolo (T_A+T_{EG}) é limitado e conhecido, bem como o tempo máximo que, uma mensagem espera, a partir de sua difusão, para ser engajada. Este tempo de latência da mensagem, para a pior situação e considerando $T_A = (L/L_{min}\ T_{EG})$, tende a $(N+2)T_{EG}$, em um grupo com N participantes onde $N \leq K \leq (3N-2)$ e T_{EG} é o tempo de uma execução de engajamento. Como T_{EG} varia linearmente com o número de mensagens-protocolo trocadas na execução de engajamento, a latência máxima de uma mensagem na pior das hipóteses é $O(N^2)$.

Um dos principais aspectos colocados, quando da elaboração desta proposta de protocolo, era tentar apresentar custos de mensagem-protocolo por mensagem engajada baixos. Este aspecto, para sistemas em tempo real, é importante já que, uma vez impedindo sobrecargas de mensagens de controle, o meio de comunicação ficaria mais disponível para mensagens de aplicação (inclusive para comunicação ponto a ponto), tão importantes para os atendimentos de tempos de respostas e deadlines. Neste sentido, protocolos como o de [Luan 90] e [Melliar-Smith 90] serviram de modelo. Assim, o engajamento se faz com L mensagens por vez e os custos de mensagens de controle de cada engajamento são distribuídas entre estas L mensagens. Em vista disto, quanto maior o L, menor é o overhead apresentado pelo algoritmo.

O protocolo proposto é limitado, embora assíncrono. Seu desempenho foi apresentado no item anterior, na forma de latência das mensagens, para diferentes situações. O uso deste protocolo em uma rede local tipo Ethernet, conforme os valores calculados para a latência no item anterior (50 [ms] a 392 [ms]), demonstram a aplicabilidade deste protocolo em aplicações de tempo real, onde os tempos de resposta e deadlines envolvidos não sejam inferiores ao limite máximo da latência.

A escolha da forma de ordenação no engajamento das L mensagens, foi no sentido da flexibilização, evitando a utilização de mecanismos globais criados apartir de recursos distribuídos para a obtenção de uma ordenação total estrita. Evitando com isto os custos no sentido da manutenção da coerência nas ordenações obtidas nestes meios, bastantes susceptíveis aos atrasos variáveis e baixa confiabilidade dos suportes de comunicação. Por isto a ordenação proposta, baseada em ordenações locais e na disposição das estações emissoras no anel virtual, é mais barata em termos de custos de mensagens e material necessário quando comparado com outras técnicas utilizadas (sincronização de relógios, técnicas de consenso, etc ...)

O protocolo foi projetado no sentido de tolerar a presença de faltas de omissão e crash. O acordo neste protocolo é obtido com a maioria dos participantes corretos. O tratamento de faltas neste protocolo (recuperação de commits) não implica em alterações da evolução normal do algoritmo para estações

corretas. As estações em processo de recuperação do estado de commit, executam os engajamentos perdidos em paralelo com a execução atual do protocolo para as demais. Este ponto é bastante importante se comparado com protocolos como o de [Birman 87] e de [Chang 84] onde aspectos de detecção, tratamento e recuperação de excessões implicam em acrescimos significativos no tempo de execução do protocolo. O protocolo proposto a exemplo de [Chang 84] necessita de reconfiguração do anel quando a estação possuidora do token apresenta uma falta de "crash". Porém os custos envolvidos na reconfiguração são bem menos expressivos no proposto.

4.6 - Comparação do protocolo proposto com o estudo bibliografico

Como apresentado no capítulo anterior, um critério para a escolha de um protocolo são os aspectos de custos nele envolvidos. Neste sentido, os protocolos apresentados no capítulo dois serão analisados quanto a seus custos de overhead e latência, e comparados com o proposto. Com a finalidade de comparação das latências dos vários protocolos apresentados neste texto, condições idênticas de carga e de limites de tempos serão tentados para facilitar a obtenção de conclusões com estas comparações. Neste sentido, os limites de tempo ou timeouts dos diferentes protocolos levarão em consideração as características de seus protocolos, mas apresentarão um fator de multiplicação (σ) da mesma ordem. As condições de engajamento (número de mensagens) serão assumidas como as mesmas.

Para o protocolo proposto a latência máxima de uma mensagem, para L=NK com K=1 e σ = 3N/(K+1) = 1,5N, será de duas execuções consecutivas do protocolo, de forma que :

$$Latência = \left(\frac{N + 4}{2}\right) T_{EG} = 1{,}5N(N + 4)\delta$$

o que corresponde a uma ordem de grandeza $O(N^2)$. O custo de mensagens de controle por mensagem difundida (overhead) é mensagens-protocolos/L, ou seja, para as condições acima, é 3.

O protocolo proposto apresenta abordagem próxima das de [Chang 84] e de [Luan 90]. Em semelhança ao [Chang 84] o protocolo proposto apresenta uma abordagem token-ring, porém menos sensitiva a faltas, permitindo que o sistema mude dinamicamente (frequentes falhas e/ou rápidos reparos) sem a necessidade de constantes reconfigurações do anel. Neste sentido, uma estação só vem a ser considerada faltosa após M sucessivas faltas por omissão. Outra vantagem apresentada é que, para um sistema R-recuperado ("System R-resilient"), o protocolo proposto necessita apenas de uma transferência de token (uma execução) para que as mensagens sejam engajadas, enquanto que em [Chang 84] são necessárias R transferências de token (R execuções).

Desta forma a latência máxima de uma mensagem em [Chang 84] será RT, onde T é o tempo de permanência do token na estação. Afim de comparação, será assumido que T seja o tempo para o recebimento e reconhecimento de L mensagens difundidas. Assim tem-se:

T = Tempo de espera pelas L mensagens + Tempo para envio dos L N reconhecimentos

onde: Tempo para envio dos N reconhecimentos = Nδ e

$$\text{Tempo de espera pelas L mensagens} = \alpha L\delta$$

o tempo de espera pelas L mensagens, como no algoritmo proposto, deve levar em consideração a intermitência nas características das comunicações da aplicação. Para este caso, este tempo de espera também será assumido como proporcional ao número de L mensagens, multiplicando por um fator α. Para efeito de comparação este fator será assumido como $\alpha = \sigma$. Assim, para $L = NK$ e $K = 1$, tem-se:

$$\alpha = \sigma = 3N/(K+1) = 1{,}5\ N$$

$$\text{Tempo de espera pelas L mensagens} = 1{,}5\ N\delta N = 1{,}5\ N^2\delta$$

de forma que o tempo de permanencia do token será então:

$$T = 1{,}5\ N^2\delta + N^2\delta = 2{,}5\ N^2\delta$$

implicando em:

$$\textit{Latência} = 2{,}5RN^2\delta$$

A ordem de grandeza da latência do protocolo de [Chang 84], para $L = NK$, $K = 1$ e $R << N$, é a mesma do protocolo proposto, isto é, $O(N^2)$. Esta grandeza cresce proporcionalmente ao fator R de "recuperação" ("resiliency"), chegando a $O(N^3)$ para $R \cong N$. O número de mensagens de controle (overhead) no protocolo de [Chang 84] será $NL + 2R$ mensagens-protocolo por mensagem difundida para um sistema subutilizado (a transferência de token não é enviada por carona) e igual a NL para um sistema sobrecarregado, onde o token é enviado por carona. Ou seja, seu overhead é diretamente proporcional ao número de mensagens engajadas, enquanto o do proposto é inversamente proporcional.

Uma outra abordagem próxima a proposta é a apresentada por [Luan 90]. Neste algoritmo o número de mensagens engajadas por execução se aproxima da proposta. O seu convite (pedido-de-commit) é enviado por qualquer estação que não tenha engajado suas mensagens em um certo tempo ou que possua seu buffer de mensagens a engajar cheio. Porém as mensagens a engajar serão determinadas em consenso com as demais estações que responderem ao convite. Isto provoca uma "round" a mais, que o protocolo proposto, de trocas de mensagens entre as participantes. Ademais, em [Luan 90] uma estação está habilitada a recuperar estados de commit através de suas memórias não voláteis além de armazenar, por tempo indeterminado suas mensagens afim de garantir que todas as estações as recebam, o que leva a um alto custo implícito, indesejável (e as vezes inviável) para a realidade do protocolo proposto neste trabalho.

A latência de uma mensagem no protocolo de [Luan 90] dependerá da sua ocorrência, ou não, no buffer de recepção de todas estações "cohort", ou seja, uma mensagem que não esteja na maioria $(N/2 + R)$ das estações não será engajada até isto ocorrer. Considerando a não existência de faltas, de forma que todos os buffers possuam, com alta probabilidade, as mensagens em no máximo duas execuções consecutivas, tem-se para $L = NK$ e $K = 1$.

$$\text{Latência} = 2\ (T_{\text{espera das mensagens}} + T_{\text{execução do protocolo}})$$

$$T_{\text{execução do protocolo}} = \text{Número de mensagens-protocolo}\ \delta = (4N+2R)\delta$$

$$T_{\text{espera das mensagens}} = \alpha\ L\delta = 1{,}5\ N^2\delta$$

$$\textit{Latência} = (3N^2 + 8N + 4R)\delta$$

Assim, a latência é da ordem de grandeza $O(N^2)$ como o protoclo proposto. Em relação ao overhead envolvido, o protocolo de [Luan 90] apresenta :

$$\frac{4N + 2R}{N} \cong 4 \quad \text{[mensagens-protocolo/mensagem difundida]}$$

para R<<N.

O ABCAST de [Birman 87] possui a coordenação distribuída de forma que cada estação é responsável em dar a ordem total e o acordo sobre as suas mensagens difundidas. Com isto, a ordenação global de mensagens de distintas estações emissoras é conduzida por distintas coordenadoras, aumentando o desempenho. As mensagens difundidas são colocadas no buffer de recepção conforme a ordem de chegada e reordenadas a medida que se obtem os seus respectivos timestamps finais. Porém, se uma falta ocorre em uma estação E_s cuja mensagem $m_{s,r}$ está presente neste buffer e com timestamp menor que o de uma outra $m_{i,k}$ a mensagem $m_{i,k}$, após receber o timestamp final necessita esperar a detecção da falta e decisão do timestamp final da $m_{s,r}$ antes de ser engajada.

A latência máxima de uma mensagem para o ABCAST de [Birman 87], sob condições normais (sem faltas), é dado por:

$$\text{Latência} = \text{Tempo espera pelos N timestamps} + \text{Tempo difusão do timestamp final}$$

$$\text{Tempo difusão do timestamp final} = N\delta$$

o problema de recepção dos N timestamps é bem mais crítico neste protocolo que no proposto, visto que no proposto é feita a decisão por maioria enquanto que em [Birman 87] a decisão se dá com todas as estações operacionais do sistema. Em vista disto talvez fosse mais conveniente que $\alpha > \sigma$. Porém, como foi discutido no inicio desta secção, o objetivo é apenas de comparação entre estes protocolos de forma que será arbitrado, mesmo assim, $\alpha = \sigma$. Logo:

$$\text{Tempo espera pelos N timestamps} = \alpha N\delta = 1{,}5\ N^2\delta$$

$$\textit{Latência} = 1{,}5N^2\delta + N\delta = (1{,}5N+1)N\delta$$

o que implica na mesma ordem de grandeza ($O(N^2)$) do protocolo proposto. O overhead do protocolo de [Biman 87] é $2NL = 2N^2$, ou seja seu custo operacional é extremamente alto quando o suporte de comunicação apresenta carga elevada.

A latência de uma mensagem para [Melliar-Smith 90] depende do número de mensagens de aplicação que a seguem, emitidas por estações distintas. O número mínimo de mensagens de aplicação, de estações distintas, necessárias para a mensagem ser engajada é $(N + R + 1)/2$. No entanto, a recepção das mensagens é probabilista, não havendo nenhuma garantia que este número mínimo de mensagens irá existir, o que não é aconselhável para aplicações em tempo real. As estações (não faltosas) quando não produzem mensagens por um certo tempo, enviam uma mensagem nula para o reconhecimento das mensagens recebidas e ainda não reconhecidas pelo sistema. A latência máxima de uma mensagem pode ser dada, então, em função deste tempo máximo para envio de uma mensagem por uma estação, de forma que:

$$Latência = (3N^2 + 8N + 4R)\delta$$

Assim, a latência é da ordem de grandeza $O(N^2)$ como o protoclo proposto. Em relação ao overhead envolvido, o protocolo de [Luan 90] apresenta :

$$\frac{4N + 2R}{N} \cong 4 \quad \text{[mensagens-protocolo/mensagem difundida]}$$

para R<<N.

O ABCAST de [Birman 87] possui a coordenação distribuída de forma que cada estação é responsável em dar a ordem total e o acordo sobre as suas mensagens difundidas. Com isto, a ordenação global de mensagens de distintas estações emissoras é conduzida por distintas coordenadoras, aumentando o desempenho. As mensagens difundidas são colocadas no buffer de recepção conforme a ordem de chegada e reordenadas a medida que se obtem os seus respectivos timestamps finais. Porém, se uma falta ocorre em uma estação E_s cuja mensagem $m_{s,r}$ está presente neste buffer e com timestamp menor que o de uma outra $m_{i,k}$ a mensagem $m_{i,k}$, após receber o timestamp final necessita esperar a detecção da falta e decisão do timestamp final da $m_{s,r}$ antes de ser engajada.

A latência máxima de uma mensagem para o ABCAST de [Birman 87], sob condições normais (sem faltas), é dado por:

$$\text{Latência} = \text{Tempo espera pelos N timestamps} + \text{Tempo difusão do timestamp final}$$

$$\text{Tempo difusão do timestamp final} = N\delta$$

o problema de recepção dos N timestamps é bem mais crítico neste protocolo que no proposto, visto que no proposto é feita a decisão por maioria enquanto que em [Birman 87] a decisão se dá com todas as estações operacionais do sistema. Em vista disto talvez fosse mais conveniente que $\alpha > \sigma$. Porém, como foi discutido no inicio desta secção, o objetivo é apenas de comparação entre estes protocolos de forma que será arbitrado, mesmo assim, $\alpha = \sigma$. Logo:

$$\text{Tempo espera pelos N timestamps} = \alpha N\delta = 1{,}5\ N^2\delta$$

$$Latência = 1{,}5N^2\delta + N\delta = (1{,}5N+1)N\delta$$

o que implica na mesma ordem de grandeza ($O(N^2)$) do protocolo proposto. O overhead do protocolo de [Biman 87] é $2NL = 2N^2$, ou seja seu custo operacional é extremamente alto quando o suporte de comunicação apresenta carga elevada.

A latência de uma mensagem para [Melliar-Smith 90] depende do número de mensagens de aplicação que a seguem, emitidas por estações distintas. O número mínimo de mensagens de aplicação, de estações distintas, necessárias para a mensagem ser engajada é $(N + R + 1)/2$. No entanto, a recepção das mensagens é probabilista, não havendo nenhuma garantia que este número mínimo de mensagens irá existir, o que não é aconselhável para aplicações em tempo real. As estações (não faltosas) quando não produzem mensagens por um certo tempo, enviam uma mensagem nula para o reconhecimento das mensagens recebidas e ainda não reconhecidas pelo sistema. A latência máxima de uma mensagem pode ser dada, então, em função deste tempo máximo para envio de uma mensagem por uma estação, de forma que:

$$Latência = \left(\frac{N + R + 1}{2}\right) T_{imeout}$$

onde Timeout é o tempo limite para a difusão de uma mensagem, de modo que, quando ultrapassado sem que nenhuma mensagem da aplicação tenha sido enviada, uma mensagem nula deve ser enviada para o reconhecimento. Como [Melliar-Smith 90] nada apresenta a respeito da ordem deste timeout será assumido que:

$$T_{imeout} = \sigma\delta = 1{,}5\ N\delta$$

$$Latência = \left(\frac{N + R + 1}{4}\right) 3N\delta$$

a latência, como os anteriores, apresenta ordem de grandeza $O(N^2)$. O overhead envolvido no protocolo tende a zero para sistemas sobrecarregados, onde o reconhecimento é dado por carona e aumenta em sistemas subutilizados onde é necessária a criação de mensagens nulas para levar o ACK. Assim, o overhead máximo será o número mínimo de mensagens de diferentes estações para o seu engajamento. No entanto por suas características o overhead é considerado aleatório.

A latência do protocolo tolerante a faltas por omissão apresentado por [Cristian 85] é dada pelo valor de Δ. Assim:

$$Latência = \Delta = R\delta + (N - 1)\delta + \varepsilon$$

Como exposto no capítulo dois, o overhead é nulo, embora apresente a dificuldade inerente à sincronização dos relógios. A necessidade de redifusão das mensagens acarreta uma alta sobrecarga

L=NK K=1	Protocolo proposto	Chang	Cristian	Birman	Gligor	Melliar-Smith
Overhead [1]	3	[N, N+2]	0	2N	4	$\left[0, \left(\frac{N + R + 1}{2}\right)\right]$
Latência[2]	$\frac{N + 4}{2}\ 3N\delta$	$2{,}5RN^2\delta$	$(R+N-1)\delta+\varepsilon$	$(1{,}5N + 1)\ N\delta$	$(3N^2+8N+4R)\delta$	$\left(\frac{N + R + 1}{4}\right) 3N\delta$

[1] Overhead é o número de mensagens de protocolo necessárias para a difusão de uma mensagem
[2] N é o número de estações participantes e R a resistência do sistema

Tabela 4.3 - Latência máxima dos protocolos comparados

ao sistema dificultando o envio das mensagens de real interesse (novas mensagens das aplicações) da mesma forma que um alto custo de overhead. A latência é O(N), porém, a ordem de grandeza de e não foi apresentada em [Cristian 85], mas é sabido que este valor não pode ser desconsiderado.

Em conclusão, o protocolo proposto possui um excelente overhead, mesmo quando comparado com os demais e sua latência máxima é da mesma ordem dos protocolos mais significativos apresentados na literatura, superando em desempenho, inclusive, alguns destes. A tabela 4.3 resume esta secção apresentando a latência e overhead deste protocolos. Para efeito de ilustração na tabela 4.4 são apresentados os valores de latência, assumindo N=12 e R=3 e L=N. Para uma melhor visualização as figuras 4.6 e 4.7 resumem, em forma de gráficos, os valores de overhead e latência, respectivamente, para os protocolos apresentados. É assumido L=N e R=3 para as figuras 4.6 e 4.7a. Para a figura 4.7b assumi-se R=5.

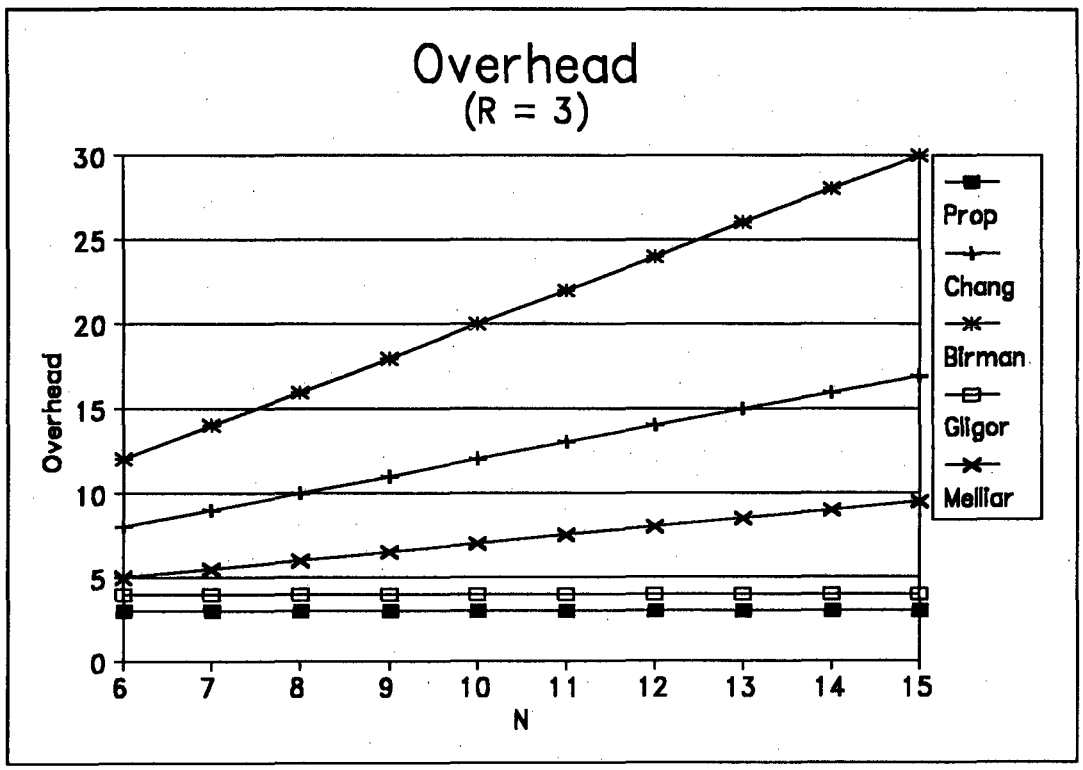
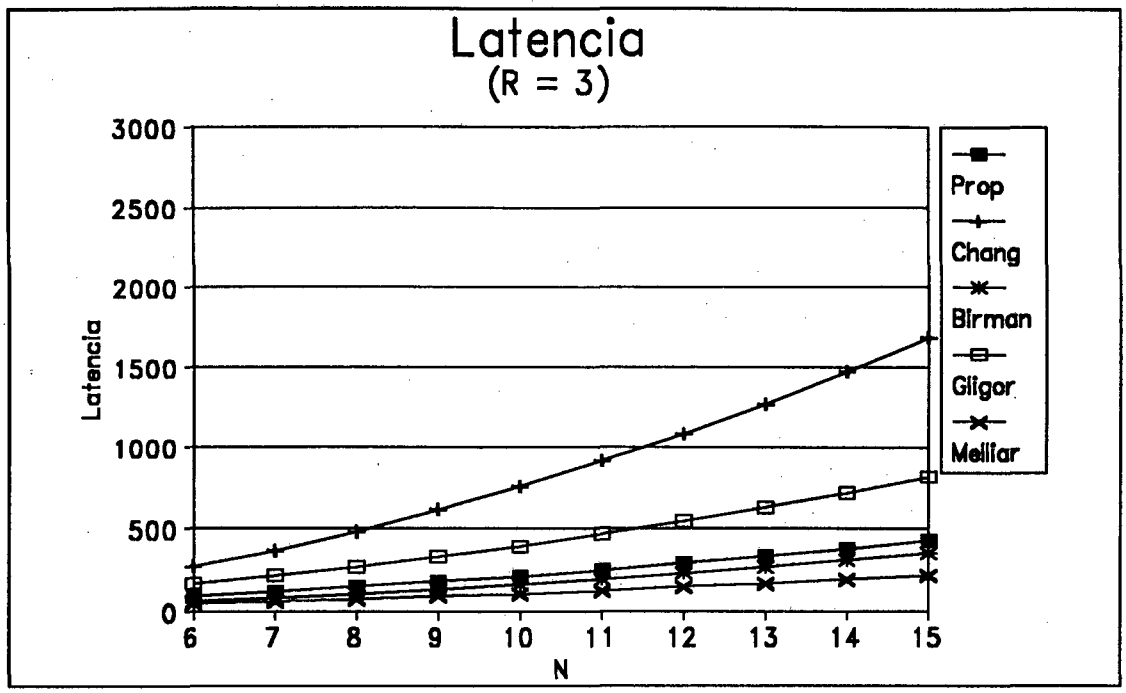
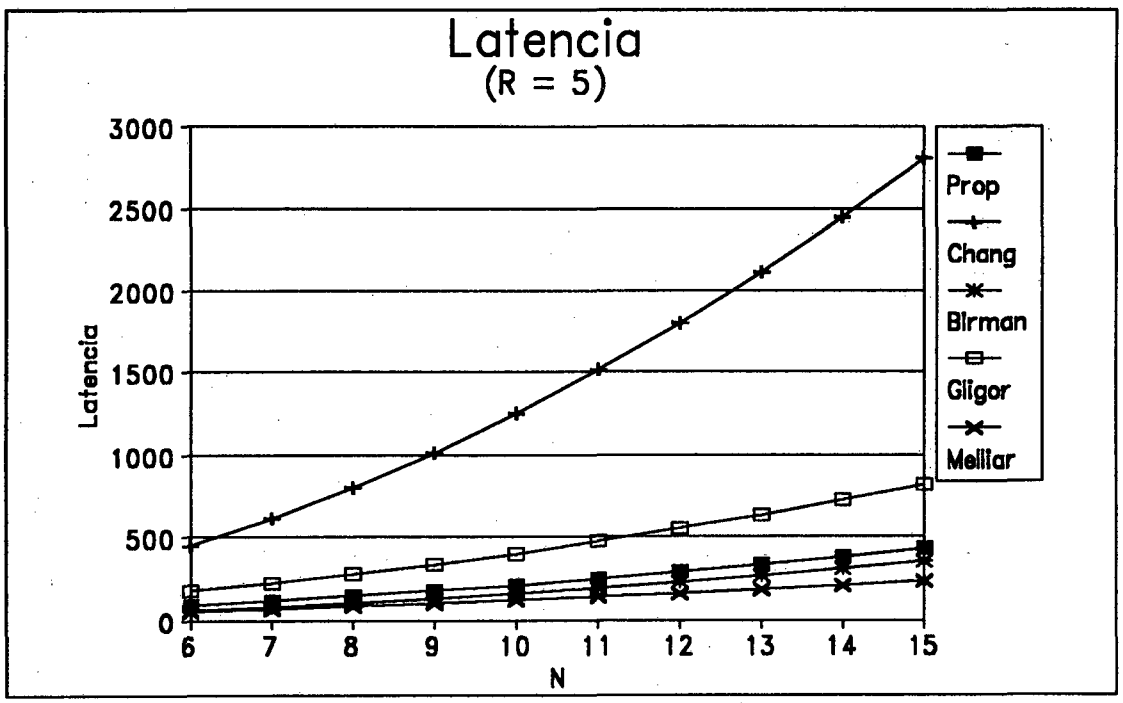


Figura 4.6 - Overhead x Número de estações, para R=3



(a) - Latência [δ] x Número de estações para R=3



(b) - Latência [δ] x Número de estações para R=5

Figura 4.7 - Latência x Número de estações

L=NK K=1	Protocolo proposto	Chang	Cristian	Birman	Gligor	Melliar-Smith
Overhead	3	[12,14]	0	24	4	[0,8]
Latência	288δ	1080δ	14δ+ε	228δ	540δ	144δ

Tabela 4.4 - Latência dos protocolos comparados, para N = 12 e R = 3

4.7 - Conclusão

Este capítulo analizou a sobrecarga das mensagens de controle e os tempos máximos para o engajamento, em função dos parâmetros K e L e do número N de estações participantes no grupo.

Foi constatada a satisfação dos objetivos preliminares da proposta de protocolo de difusão confiável. O overhead é dado na proporção inversa da capacidade de engajamento, possibilitando que os custos de controle se distribuam entre as mensagens difundidas, tornando-se desprezíveis. A latência é limitada e da ordem de grandeza $O(N^2)$, o que representa um desempenho melhor que alguns protocolos existentes na literatura e permite prever o uso do protocolo para aplicações em tempo real.

CAPÍTULO 5

SIMULAÇÃO E SEUS RESULTADOS

5.1 - Introdução

Um ponto comum na concepção de sistemas distribuídos é a necessidade de descrever de maneira completa e sem ambiguidades os vários componentes sequenciais que se executam em paralelo e cooperam entre si [Courtiat 91]. Para tal foram desenvolvidas pelo CCITT e ISO Técnicas de Descrição Formal - FDTs, como por exemplo Estelle, Lotos e SDL, que propiciam uma sintaxe bem definida à especificação.

O presente capítulo descreve os esforços feitos no sentido de validar o algoritmo proposto. O formalismo Estelle [ISO 9074] é usado então na descrição do algoritmo e a ferramenta ESTIM [Saqui 90] serve de base para execução das especificações Estelle. Alguns cenários de faltas são propostos no sentido de testar as potencialidades do algoritmo de difusão confiável.

A secção 5.2 apresenta suscintamente a FDT Estelle*, base para a validação através da ferramenta ESTIM, apresentada a seguir. Em 5.3 é apresentado o modelo de simulação e os resultados obtidos.

5.2 - Métodos e ferramentas para validação do algoritmo proposto

Uma vez criada uma especificação formal, é necessário validá-la garantindo a satisfação das propriedades propostas, sejam elas gerais (safety, livelock, performance) ou específicas (serviço oferecido = semântica do protocolo). Para tal validação, três técnicas podem ser utilizadas [Pehrson 89]:

- **Verificação**: prova formal que a especificação satisfaz certas propriedades desejáveis utilizando para isto métodos rigorosos e matematicamente justificados. Pode-se usar dois tipos diferentes de verificação:
  - Provando as propriedades individuais de uma especificação: propriedades são formuladas formalmente utilizando, por exemplo lógica modal e, em seguida, é verificado se o sistema satisfaz estas propriedades.

- Comparando duas diferentes especificações e provando que, ou elas são equivalentes, ou uma é a implementação correta da outra.

- **Simulação**: um modelo executável é desenvolvido e observado quando em execução.

- **Teste**: um protótipo, ou eventualmente o sistema final, é observado sob estímulos e a observação é comparada com a especificação.

A verificação é um método exaustivo que considera todas situações possíveis de ocorrer durante a execução do sistema especificado, enquanto que o teste e a simulação validam somente certos caminhos selecionados entre todas as execuções possíveis. Atualmente, ainda existem dificuldades de aplicar a técnica de verificação em especificações do tamanho encontrado na maioria das aplicações práticas, por isto a simulação ainda é a técnica mais utilizada em validação de especificações através de FDTs que permitem alguma forma de execução simulada.

O protocolo proposto foi descrito na FDT Estelle* e simulado por análise interativa da especificação através da ferramenta ESTIM (Estelle SimulaTor based on an Interpretative Machine) desenvolvida pelo LAAS-CNRS, na França.

O ESTIM permite também a verificação de uma especificação por abstração, o que foi tentado sem sucesso em nossa proposta devido ao seu tamanho.

5.2.1 - Estelle e Estelle*

Uma especificação Estelle visa descrever uma estrutura de comunicação de estados autônomos cujas ações internas são definidas por declarações em Pascal (com restrições e extensões). Uma especificação Estelle é constituida de um conjunto de **Módulos** que interagem (trocam mensagens) através de **Pontos de Interações** (IP). Os módulos são definidos como **tipos** (cabeçalho + corpo), podendo ser criadas várias **instâncias** de módulos de um determinado tipo. A gestão de instâncias de módulos (criação e destruição), a gestão de instâncias de pontos de interação (conexão e desconexão) e a sincronização entre instâncias de módulos (envio e recepção de interações) são disponíveis através de um **conjunto de primitivas** Estelle.

Estelle* é caracterizado essencialmente pela junção do Estelle a um mecanismo de Rendez-vous. As principais diferenças entre Estelle e Estelle* são apresentadas na tabela 5.1 . Introdução mais detalhadas aos formalismos Estelle e Estelle* podem ser encontrados em [ISO 9074] , [Budkowski 87], [Courtiat 87] e [Courtiat 91].

5.2.2- ESTIM

O ESTIM (Estelle SimulaTor based on an Interpretative Machine) [Saqui-Sannes 89] foi desenvolvido como parte do projeto ESPRIT-SEDOS no LAAS-CNRS, na França, e se encontra disponível atualmente na versão 4.01 para estações de trabalho SUN4.

O ESTIM foi escrito na linguagem funcional ML, utilizando um interpretador Estelle (GENESTIM), que após executar uma análise sintática da especificação e uma checagem dos tipos estáticos, carrega a árvore abstrata ML da especificação na entrada do seu ambiente de simulação.

Apartir da árvore abstrata ML é gerada toda a sequência de transições que podem ser disparadas, denominada "Grafo de alcançabilidade". Com o grafo de alcançabilidade completo é possivel fazer através do ESTIM a verificação do protocolo por abstração, ou verificação por projeção, como também é conhecida. Isto é, aplicar sob o grafo técnicas de redução baseadas nas reduções por equivalência, resultando em um "autômato quociente" que mostra o serviço fornecido por este protocolo.

	Estelle ISO	Estelle *
Atributo System	systemactivity systemprocess	systemactivity ----------
Atributo Module	activity process	activity ----------
Disciplina de Fila	individual queue common queue ----------	individual queue ---------- no queue
Escala de tempo	----------	irrelevante
Prioridades	calusula "priority" prioridade pai/filho	não permitida desabilitada
Clausula rendezvous	---------- ----------	ip?interação ip!interação

Tabela 5.1 - Estelle* vs Estelle [Saqui 90]

Em algumas situações é indesejável ou ainda impossível, devido a uma fila FIFO infinita em Estelle, a geração do grafo de alcançabilidade completo. Nestes casos, a simulação explora caminhos particulares do grafo. É fornecido para isto, em cada estado, o conjunto de transições disparáveis neste estado. As transições podem ser disparadas de duas maneiras: na primeira é o próprio usuário quem seleciona a transição desejada e na segunda, a seleção é feita pelo simulador, que com técnicas de seleção aleatória determina uma sequência de transições tomando como base a quantidade de transições desejadas pelo usuário. A figura 5.1 sintetiza este ambiente

5.3- Especificação do protocolo proposto

Afim de fornecer uma especificação formal do protocolo proposto fez-se uso então do FDT Estelle*, utilizando para tal, unicamente as características do padrão Estelle. A escolha do Estelle* foi, como dito anteriormente, pela possibilidade de, posteriormente, se validar o protocolo através de uma simulação com a ferramenta ESTIM.

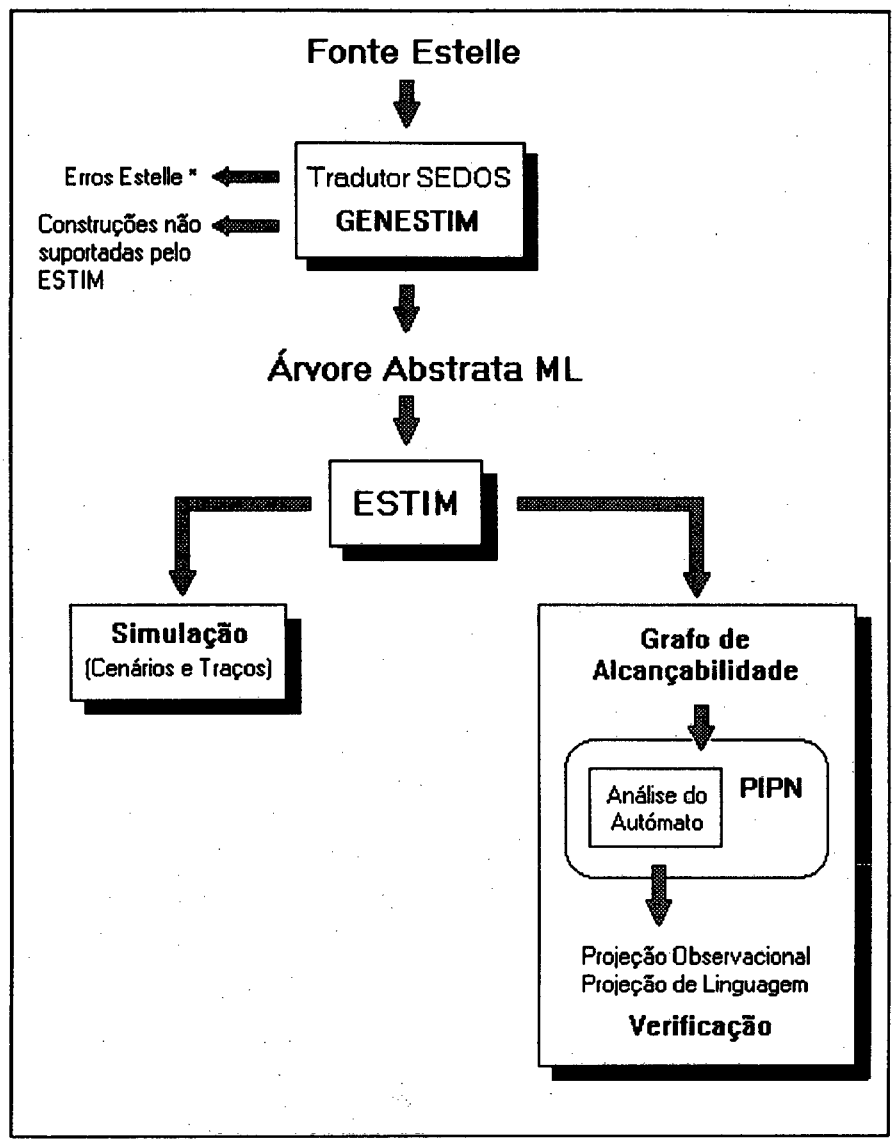


Figura 5.1 - Ambiente ESTIM [Saqui 90]

Para esta especificação, o protocolo foi dividido em módulos, de acordo com a afinidade das tarefas executadas. A estrutura do modelo de simulação do algoritmo é apresentada na figura 5.2, na figura 5.3 os módulos propostos são apresentados na forma de uma estrutura hierárquica Estelle. No apêndice B são apresentados os pseudocódigos referentes a esta especificação.

A seguir serão apresentados suscintamente as funcionalidades de cada módulo da figura 5.2.

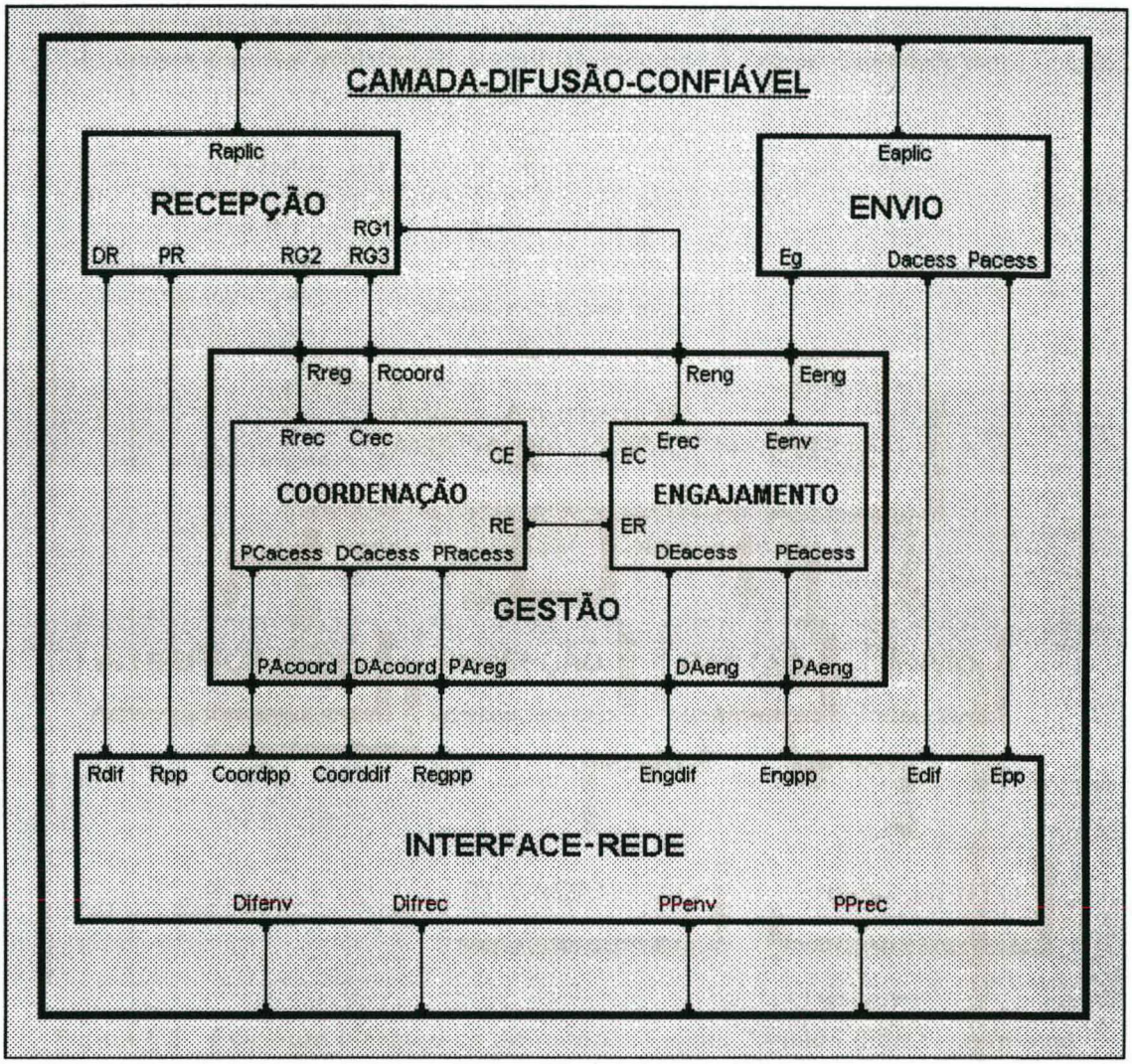


Figura 5.2 - Especificação Estelle do protocolo de difusão proposto, na camada de difusão confiável de uma estação do grupo.

5.3.1 - Módulo Estelle Recepção

O Módulo Estelle Recepção, além de ser responsável pela Recepção das mensagens difundidas, como retratado em 3.3.2.2, deve:

- **Contabilizar** o número total de mensagens nas F_i. Na totalização de L mensagens no conjunto dos F_i, se a estação for a coordenadora, enviar mensagem contendo a $L_{me}(j)$ e a nova coordenadora proposta ao módulo Estelle Coordenação. O envio desta mensagem também pode ocorrer a pedido do módulo Estelle Coordenação.

- **Verificar**, durante a submissão de um pedido-de-commit(j) sob controle do módulo Estelle Engajamento, a existência de todas mensagens de $L_{me}(j)$ fazendo-lhe o devido reconhecimento desta lista. Em caso de detecção de perda de mensagens é responsabilidade desta função iniciar o processo para recupera-las.

- **Fazer Engajamento** (atual ou perdido) das mensagens, enviando-as ao módulo Estelle Aplicação, bem como ao módulo Estelle Coordenação quando a estação pertencer ao Grupo-R.

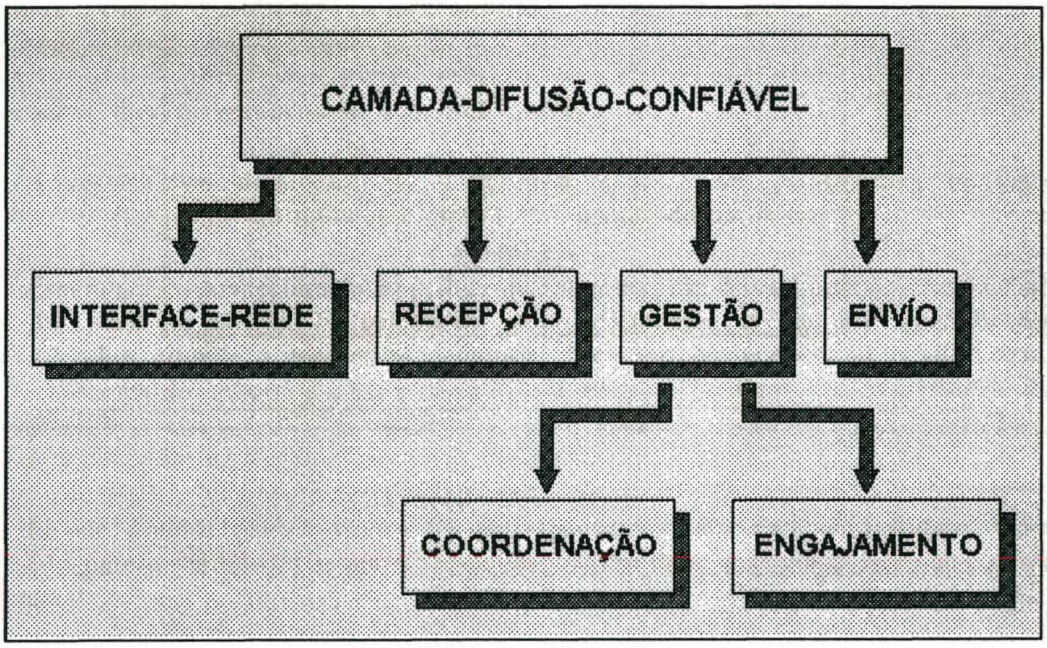


Figura 5.3 - Estrutura hierarquica do módulo Estelle Camada-difusão-confiável

5.3.2 - Módulo Estelle Envio

O módulo Estelle Envio é responsável pelas funções de recepção da aplicação de uma mensagem e a sua posterior difusão sobre a rede. Este módulo exerce o controle de fluxo sobre a ação da aplicação e tem suas "difusões"sobre a rede controladas pela janela de tamanho K.

5.3.3 - Módulo Estelle Gestão

O módulo Estelle Gestão, para melhor executar suas funções as divide em três blocos: Regeneração, Coordenação e Engajamento. As funções de Regeneração e Coordenação de commit são

agrupadas no módulo Estelle Coordenação, enquanto que as funções de participação em Engajamento de "Commit" são agrupadas no módulo Estelle Engajamento.

5.3.3.1 - Módulo Estelle Coordenação

O módulo Estelle Coordenação é responsável pelas funções de Coordenação da execução de protocolo e da regeneração de commits perdidos por estações ainda pertencentes a G_p. Outra função deste módulo é a reinserção de estações não mais pertencentes a G_p. O módulo Estelle Coordenação só é ativo em uma estação, enquanto a estação é a coordenadora e, se tornando praticamente inoperante em outras execuções onde a estação não é coordenadora.

As ações, ditas de coordenação, são então as seguintes:

- Ao receber aviso do módulo Estelle Recepção de que foram completadas L mensagens nas F_i, ou, ao estouro de um timeout, é difundido o pedido-de-commit(j).

- Espera pelo reconhecimento ao pedido-de-commit(j) das demais estações. Se receber somente ACKs até um timeout, T_E, calcula os campos da mensagem "commit", apresentados no item 3.3.2.3 e difunde a mensagem "commit". No caso de receber um NACK, difunde uma mensagem de "Abort" e tenta recuperar as mensagens perdidas através da função Recepção.

Já as ações, ditas de Regeneração, são as seguintes:

- Enviar "commit" perdido as estações que o requisitaram através de um Pedido-de-Recuperação-de-Commit-Perdido ou de um ACK de um commit imediatamente anterior ao commit perdido atual.

- Gestionar os grupos G_r (Grupo de estações em recuperação) e G_A (Grupo de estações ausentes) associados a "commits" presentes na Fila-M-msg-commit.

- Armazenar e gerenciar as mensagens "commit" na Fila-M-msg-commit e também, se a estação pertencer ao Grupo-R de um commit específico, as mensagens de aplicação correspondentes na Lista-de-Regeneração.

5.3.3.2 - Módulo Estelle Engajamento

O módulo Estelle Engajamento é responsável pelas funções ditas de Engajamento, isto é:

- Reconhecer, positiva ou negativamente, o pedido-de-commit(j). Devendo, para isto, cooperar com o módulo Estelle Recepção, responsável pela verificação da existência das mensagens que serão engajadas na execução de protocolo correspondente.

- Enviar pedido-de-recuperação-de-commit-perdido à coordenadora, em caso de detecção de perda de commit.

- Realizar "commit", atual ou perdido, enviando lista de mensagens a engajar ao módulo Estelle Recepção e a mensagem "commit" para ser armazenada no módulo Estelle Coordenação. Se a estação pertencer a G_L, o "commit" também será sinalizado ao módulo Estelle Envio através do módulo Estelle Engajamento.

- Requisitar reinserção ao anel, quando acionadas primitivas de operador para que a estação seja reinserida em G_p.

Em outras palavras, o módulo Estelle Engajamento é o responsável por realizar o commit atual e/ou perdido, em cada estação pertencente a G_p.

5.3.4 - Módulo Estelle Interface-Rede

O módulo Estelle Interface-Rede é o responsável pela interface entre a camada-difusão-confiável e o suporte de comunicação (figura 5.4). Assim, por este módulo passam todas mensagens enviadas de ou para a camada inferior.

O módulo Estelle Interface-Rede (visto na figura 5.2) é o responsável pela padronização da mensagem enviada da camada-difusão-confiável para o módulo suporte de comunicação, bem como no sentido inverso.

5.4 - Validação do protocolo proposto

O objetivo desta etapa é validar a especificação do algoritmo proposto, garantindo que esta não apresenta erros em relação ao serviço desejado, além de validar as propriedades básicas de um protocolo de difusão confiável (apresentadas no capítulo 2), no algoritmo proposto.

Os principais pontos a serem observados são apresentados a seguir:

1) Recepção, em todas as camadas de difusão confiável de estações ditas operacionais, das mensagens de aplicação difundidas por uma estação.

2) Engajamento de mensagens durante uma execução de protocolo, mantendo as propriedades estipuladas para o algoritmo.

3) Capacidade do algoritmo de detectar e recuperar corretamente as exceções ocorridas durante a evolução da aplicação .

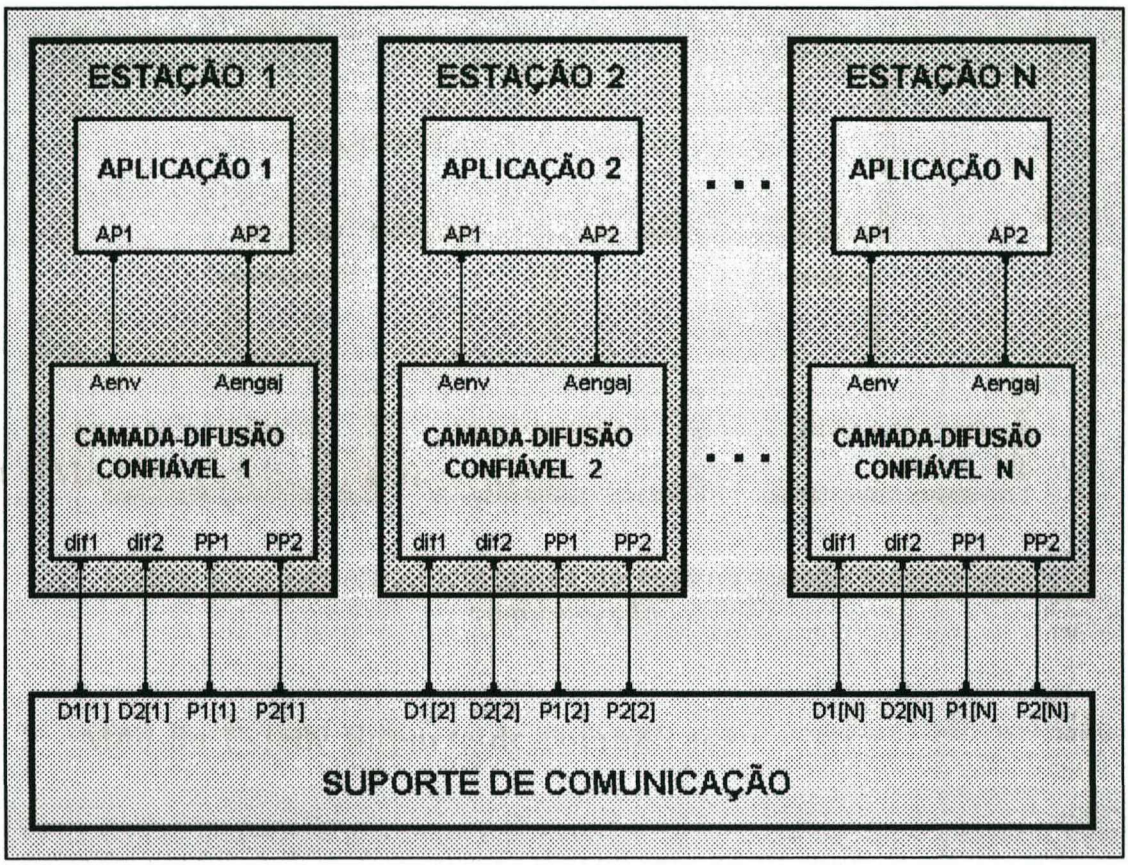


Figura 5.4 - Especificação Estelle do ambiente para simulação

5.4.1 - Especificação do ambiente para simulação

Na figura 5.4 é apresentado o ambiente proposto para a verificação destes pontos. Devido a restrição de memória do ambiente de simulação, o número de instâncias de módulo Estelle Aplicação e, consequentemente, do número de instâncias de módulo Estelle Camada-difusão-confiável, variaram com a complexidade e as necessidades do ponto a ser verificado. Para pontos menos complexos como recepção e recuperação de mensagens foram utilizadas três instâncias de estações (Aplicação + Camada-difusão-confiável). Para itens mais complexos, foram utilizadas de cinco a oito estações.

No apêndice C é encontrado o pseudocódigo da especificação do ambiente para simulação. As funcionalidades de cada módulo são descritas a seguir:

5.4.1.1 - Módulo Estelle Aplicação

O módulo Estelle aplicação é bastante simplificado e representa unicamente o envio e a recepção de mensagens difundidas para ou de outras aplicações, respectivamente.

Figura 5.5 - Estrutura hierarquica do Módulo Estelle Suporte-de-comunicação

5.4.1.2 - Módulo Estelle Suporte-de-comunicação

O módulo Estelle Suporte-de-comunicação representa o encaminhamento de uma mensagem ao seu destino, permitindo a simulação de sucesso ou fracasso nesta transferência de mensagens. A figura 5.5 apresenta a estrutura hierárquica deste módulo, subdividido-o em serviço-Multiponto, serviço-Ponto-a-ponto e serviço-Perturbação responsáveis pelas funções de serviços de comunicação e da sinalização para ocorrências de erros nas comunicações respectivamente.

Módulo Estelle Ponto-a-ponto

O módulo Estelle Ponto-a-ponto é responsável pela comunicação Ponto a ponto das estações. Isto é, comunicação de uma certa estação para uma outra pré-determinada.

Módulo Estelle Perturbação

O módulo Estelle Perturbação tem a função de sinalizar ao módulo Estelle Multiponto, através de uma mensagem denominada "mensagem-perda", os tipos de faltas que devem ocorrer. A mensagem-perda enviada pode sinalizar:

- Perda de mensagens da aplicação,
- Perda de mensagens de pedido-de-commit,
- Perda de mensagens commit,
- Perda de mensagens de pedido-de-commit e commit.

Esta mensagem-perda indica também qual o grupo de estações que irão sofrer a perda. Tais grupos são denominados no apêndice C de Grupo-site e podem conter de uma a N/2 estações.

Módulo Estelle Multiponto

O módulo Estelle Multiponto tem a função de difundir as mensagens recebidas do módulo Estelle Camada-difusão-confiável a todas estações pertencentes a G_p. Ou seja, enviar as mensagens recebidas em um dos N pontos de interação D1 (figura 5.4) aos pontos de interação D2 correspondentes as estações pertencentes a G_p.

Este módulo também é responsável por simular a ocorrência de faltas no sistema, não enviando a mensagem sinalizada pela mensagem-perda recebida do módulo Estelle Perturbação à(s) estação(ões) por esta indicada. Assim, o módulo Estelle Multiponto, ao difundir a mensagem especificada pela mensagem-perda, o deve fazer somente às estações pertencentes a G_p e não pertencentes ao Grupo-site.

5.4.2 - Resultados de simulação

Através da simulação foram explorados caminhos particulares do grafo de alcançabilidade, permitindo a observação da conformidade da especificação do protocolo proposto com o serviço desejado, nos pontos apresentados em 5.4.

Para tal simulação foram utilizados quatro modelos de sistema, os quais são apresentados na tabela 5.2 com as respectivas sinalizações de faltas aos sistemas. Tais sinalizações implicam em diferentes situações de funcionamento do serviço de difusão confiável.

Modelo 1

Este modelo se apresenta com o módulo Estelle Perturbação desativado. O sistema possui três instâncias de estações (N=3), capacidade de engajamento L = 3, grau de omissão M = 2, grau de resistência R = 2 e filas F_i de capacidade K = 2.

Através deste modelo foi simulada a condição normal de uma execução de protocolo, isto é, não houve perdas de mensagens difundidas, mensagens de pedido-de-commit e/ou mensagens de commit.

Modelo 2

Para este modelo também foi criado um sistema com três instâncias de estações (N=3), capacidade de engajamento L = 3, grau de omissão M = 2, grau de resistência R = 2 e filas F_i de capacidade K = 2, porém, com o módulo Estelle Perturbação ativado.

Diante deste modelo de sistema foram observadas as sequintes situações:

Situação 2.1 - Módulo Estelle Perturbação sinalizando perda de mensagens difundidas

Nesta situação foram simulados os seguintes cenários de exceções:

1) Perda de mensagens pela coordenadora, causando o envio de um $L_{me}(j)$ não "apropriado".

2) Perda de mensagens por todas estações de forma a não completar a capacidade de engajamento L no tempo T_A.

3) Detecção da perda de mensagens de uma estação através da sua próxima emissão, tal que a detecção ocorre antes do respectivo pedido-de-commit. Foi simulada também a perda, simultânea, de mensagens de várias estações .

4) Detecção da perda de mensagens através do pedido-de-commit. Neste item foram simulados a recuperação da mensagem antes e depois de terminar o tempo de espera aos reconhecimentos de pedido-de-commit (T_E) dado pela coordenadora. Foi admitido a estação, quando recuperada a mensagem antes de estourar T_E ainda participar da referente execução de engajamento. No entanto, quando a mensagem foi recebida após o timeout de espera T_E, a estação foi considerada pertencente a G_A (Grupo-Ausentes).

Situação 2.2 - Módulo Estelle Perturbação sinalizando Perda de Pedido-de-commit

Nesta situação foi simulado a não formação de um "commit quorun" para o engajamento das mensagens provocando a perda do pedido-de-commit por duas estações do grupo.

Modelo 3

Este modelo representa um sistema com cinco instâncias de estações (N = 5), capacidade de engajamento L = 3, grau de omissão M = 3, grau de resistência R = 3 e filas F_i de capacidade K = 2, no qual foi ativado o módulo Perturbação para sinalizar Perda de pedido-de-commit e commit com a detecção de perda do commit feita na recepção do pedido-de-commit posterior. Esta situação teve o objetivo de reforçar os resultados encontrados na situação 2.2, item 4. Também foi ativado o módulo Estelle Perturbação para sinalizar Perda de commit, tendo recebido o Pedido-de-commit.

Modelo 4

Para este modelo foi utilizado um sistema com oito instâncias de estações (N = 8), capacidade de engajamento L = 4, grau de omissão M = 5, grau de resistência R = 5 e filas F_i de capacidade K = 2, no qual foi ativado o módulo Perturbação para sinalizar sucessivamente perda de X mensagens difundidas, e perda de mensagens de pedido-de-commit e commit

Modelo	Status do Módulo Perturbação	Situaçoes Simuladas
Modelo 1	Módulo Perturbação desativado	Engajamento sob condições normais com formação de commit quorum
Modelo 2	Módulo perturbação sinalizando Perda de mensagens difundidas pela aplicação	Estouro do timeout T_A
		Envio de L_{me} não apropriado
		Detecção da perda através da próxima mensagem
		Detecção da perda através do pedido-de-commit
	Módulo perturbação sinalizando em Perda da mensagem Pedido-de-commit	Não formação de commit quorum
Modelo 3	Módulo perturbação sinalizando Perda da mensagens Pedido-de-commit e commit	Perda de uma execução de engajamento
	Módulo perturbação sinalizando Perda da mensagem commit	Participação do commit quorun com perda de commit
Modelo 4	Modulo perturbação sinalizando perda sucessiva de mensagens e "Perda de mensagem Pedido-de-commit e commit"	Perda de até M execuções de engajamento
		Perda de mais de M execuções de engajamento

Tabela 5.2 - Situações Simuladas

Neste modelo simulou-se constantes perdas dos três tipos de mensagens descritos acima de forma a se observar o comportamento diante de faltas sucessivas. O número de perdas, sem detecção, variou de forma a cobrir todo o espectro do grau de omissão M, isto é, a detecção ocorreu desde a primeira perda até a estação ser considerada faltosa e inserida em G_f(inexistência de detecção).

Em todos estes modelos apresentados, foram explorados (de forma selecionada pelo usuário e/ou aleatória) inúmeros caminhos do grafo de alcançabilidade, obtendo sempre o resultado esperado, livre de deadlocks, recepções indefinidas e/ou loops indesejáveis.

Também foi observada a conformidade do protocolo proposto com as propriedades básicas de um protocolo de difusão confiável apresentadas no capítulo dois.

5.4.3 - Comentários

A proposta de protocolo de difusão confiável apresentada neste trabalho foi validada através de uma simulação, utilizando para tal a ferramenta ESTIM. Embora a simulação não seja um método exaustivo, não considerando desta forma todas as situações possíveis de ocorrer, esta permite percorrer inúmeros caminhos do grafo de alcançabilidade. Como foi dito anteriormente, o disparo de uma transição pode ser feito de forma aleatória ou por escolha do usuário. O disparo de forma aleatória permite observar a possível existência de estados de deadlock, parciais ou totais, bem como recepção não especificadas ou transições nunca disparadas ("safety propriety"). Já a escolha da transição a ser disparada pelo usuário é adequada para detectar possíveis "bugs" na especificação.

A ferramenta ESTIM estabelece algumas estatísticas durante a sessão de simulação permitindo assim se conhecer todas as transições disparadas, as mais requisitadas e as que nunca ocorreram na sessão. Com base nestas estatísticas pode-se forçar (através da escolha de transições via usuário) o disparo de determinadas transições, garantindo que, toda a especificação foi analizada e satisfez as propriedades desejadas, pelo menos sob as condições analisadas.

Um dos problemas encontrados na simulação foi a falta de condições de verificação de tempos, impossibilitando a confirmação do desempenho do protocolo. Esta verificação deverá ser feita quando na implementação do protocolo.

5.5 - Conclusão

Foi apresentado neste capítulo uma descrição da especificação do protocolo proposto em Estelle, utilizada na validação do algoritmo através da ferramenta ESTIM. Foram também apresentados os vários modelos de instâncias utilizados na simulação. Os pontos observados foram:

1) Recepção, em todas as camadas de difusão confiável de estações ditas operacionais, das mensagens de aplicação difundidas por uma estação.

2) Engajamento de mensagens durante uma execução de protocolo, mantendo as propriedades estipuladas para o algoritmo.

3) Capacidade do algoritmo de detectar e recuperar corretamente as exceções ocorridas durante a evolução da aplicação .

Em todos os modelos foram explorados inúmeros caminhos do grafo de alcançabilidade, obtendo sempre a conformidade da especificação com o serviço desejado.

CAPÍTULO 6

CONCLUSÃO E PERSPECTIVAS FUTURAS

Foi apresentado neste trabalho uma proposta de protocolo de difusão confiável atômico, a qual satisfaz às três propriedades básicas: acordo, ordenação e terminação, mesmo na presença de faltas de omissão e crash.

A propriedade de ordenação foi garantida através de uma ordenação total das mensagens. A forma de ordenação destas mensagens foi dada em função de suas ordens locais, nas respectivas emissoras, e a localização de tais estações no anel virtual. Esta forma de ordenação veio no sentido de se evitar os custos envolvidos com manutenção da coerência nas ordenações através de mecanismos globais, tais como, sincronização de relógios ou técnicas de consenso.

Com o engajamento de mais mensagens por execução, obteve-se um menor custo de mensagens de controle por mensagem difundida, diminuindo consideravelmente a sobrecarga que as mensagens de controle provocavam no meio, quando analisados outros protocolos da literatura.

O protocolo proposto possui terminação assincrona, mas com tempo máximo de terminação limitado e conhecido. Tal tempo máximo foi apresentado na forma de latência máxima, valores de latência para o uso do protocolo em uma rede local tipo Ethernet foram dados, de forma a ilustrar a aplicabilidade em tempo real para tempo de resposta e deadlines superiores a esta latência.

Foram também apresentados os mecanismos de faltas que permitiram que mensagens engajadas pudessem ser recuperadas por estações ausentes de menos M execuções consecutivas de protocolo, mesmo na presença de R estações faltosas. Diante desta possibilidade de recuperação de commits anteriores, é possível se minimizar, em alguns casos, os honerosos custos de transferência de estados em modelos de processamento replicados na aplicação.

Com o acordo necessitando de uma maioria e não da totalidade das estações participantes, o protocolo se tornou menos sensitivo a faltas destas estações, admitindo um maior paralelismo entre o tratamento de faltas (e recuperação de estados de commit anteriores) e a execução do commit atual pelas estações corretas.

O protocolo proposto foi apresentado na técnica de descrição formal Estelle e validado, através de simulação, pela ferramenta ESTIM. Para tal foram criados vários modelos de simulação, os quais exploraram inúmeros caminhos do grafo de alcançabilidade, obtendo sempre a conformidade da especificação com o serviço desejado. Ou seja, as propriedades de acordo, ordenação e terminação foram garantidas conforme esperado. No entanto, como a simulação não é um método de validação exaustivo,

a real potencialidade do protocolo poderá ser avaliada diante de uma aplicação real qua envolva processamento replicado.

Neste trabalho foi apresentado um estudo bibliográfico, no qual alguns dos mais relevantes protocolos de difusão confiável foram examinados em seus aspectos de custos e desempenho. Uma análise comparativa entre o desempenho destes protocolos e o proposto, apresentou resultados favoráveis a este último. Enquanto a maioria dos protocolos analisados apresentavam um crescimento linear do overhead com o número de mensagens engajadas, o proposto possibilitou que este overhead fosse distribuído entre as mensagens difundidas de forma a se tornar desprezível. Já a latência se apresentou na mesma ordem de grandeza dos demais ($O(N^2)$), mas com valores menores que alguns deles.

Este trabalho, de uma certa maneira, se encaixa na linha de projetos desenvolvidos no LCMI/UFSC no sentido de se formar suportes para a implementação de tolerância a faltas através de modelos de alta replicação de processamento. Assim, a experiência adquirida neste trabalho deve contribuir na própria definição destes modelos.

BIBLIOGRAFIA

[Azema 85] Azema P., Papapanagiotakis G., "Protocol Analysis by using Predicate Nets"; *Proc of the 5th International Workshop on Protocol Specification, Verification and Testing*, France, June 1985.

[Babaoglu 85] Babaoglu O. e Drummond R. P., "Streets of Byzantium: Network Architectures for Fast Reliable Broadcast"; *IEEE Transaction on Software Engeneering*, vol SE-11, No. 6, June 1985, pp 546 - 554.

[Babaoglu 88] Babaoglu O. , Stephenson P. e Drummond R. P. ,"Reliable Broadcast and Communication models: Tradeoffs and Lower Bounds"; *Distributed Computing*, 2, 1988, pp 177 - 189.

[Birman 87] Birman K. P. e Joseph T. A. , "Reliable Communication in the Presence of Failures"; *ACM Transaction on Computer Systems*, vol 5, No. 1, Feb 1987, pp 47 - 76.

[Bochmann 90] Bochmann G. V., "Protocol Specification for OSI"; *Computer Networks and ISDN Systems*, 18, 1989/90, pp 167 - 184.

[Budkowski 87] Budkowski S., Dembinski P., "An Introductin to Estelle : A Specification Language for Distributed Systems"; *Computer Networks and ISDN Systems 14*, 1987, pp 3 - 23.

[Cart 87] Cart M. , Ferrie J. , Mardiyanto S. , "Atomic Broadcast Protocol, Preserving Concurrency, For an Unreliable Broadcast Network"; *Local Communication Systems: LAN and PBX*, IFIP, 1987.

[Chang 84] Chang Jo-Mei , Maxemchuk N. F. , "Reliable Broadcast Protocols"; *ACM Transaction on Computer Systems*, vol 2, No. 3, August 1984, pp 251 - 273.

[Courtiat 87] Courtiat J. P. , Dembinski P., Groz R., Jard C., "Estelle : An ISO Language for Distributed Algorthms and protocols", *TSI*, mai, 1987

[Courtiat 89] Courtiat J. P. , Diaz M., "Description Formelle de Protocoles OSI en Estelle", *Seminaire Franco-Brésilien sur les Systèmese Informatiques Répartis*, 11-14/09/89, Florianópolis, SC, Brasil, pp 47- 55.

[Courtiat 91] Courtiat J. P. , Diaz M. , Mazzola V. B. , Saqui-Sannes P., "Description Formelle de Protocoles et de Services OSI en Estelle et Estelle*", *CFIP91*, Pau, Setembro 1991.

[Cristian 85] Cristian F., Aghili H., Strong R., Dolev D. , "Atomic Broadcast : From Simple Message Diffusion to Byzantine Agreement"; *In FTCS15*, Ann Arbor, USA, June 1985.

[Delta-4], "Delta-4 Architecture Guide", *Delta-4 Consortium*, Deecember 1990 (Delta-4 Document Número G90050/I1/R)

[Fernandez 88] Fernandez J. C., "ALDEBARAN : un système de verification par rèduction de processus communicantes"; Thèse de Doctorat, Université Joseph Fourier, Grenoble, mai 1988.

[Fraga 89] Fraga J S, Farines J M, Abreu WM, Silva E S, Nacamura Junior L, Coelho Filho O, "ADES: Ambiente de Desenvolvimento e Execução de Software Distribuído"; *Seminário Franco Brasileiro em Sistemas Informáticos Distribuídos*, Florianópolis, SC, Brasil, Setembro 1989, pp

[ISO 9074] "Estelle, a Formal Description Technique Based on an Extended State Transition Model"; ISO - DIS 9074, June 1897.

[Joseph] Joseph T. A. , Birman K. P. , "Reliable Broadcast Protocols", *Distributed System* , cap 14, Edited bY Sape Mullender, ACM Press, pp 293 - 317.

[Kaashoek 89] Kaashoek M F, Tanenbaum A S, Hummel S F, Bal E Henri, "An Efficient Reliable Broadcast Protocol", *ACM Operating System Review*, vol 23, Número 4, 1989, pp 5 - 19.

[Kümmerle 87] Kümmerle K, Limb J. O., Tobagi F. A., "Advances in Local Area Networks", *IEEE Press* , 1987

[Lamport 78] Lamport L. , "Time, Clocks, and the Ordering of Events in a Distributed System"; *Comunication of the ACM*, vol 21, No. 7, July 1978, pp 558 - 565.

[Lamport 82] Lamport L. , Shostak R. , Pease M., "The Byzantine Generals Problems"; *ACM Transaction on Programming Languages and Systems*, vol 4, No. 3, July 1982, pp 382 - 401.

[Lamport 85] Lamport L. , Melliar-Smith P., "Synchronizing Clocks in the Presence of Faults"; *Journal of the ACM*, 32(1), January 1985

[Luan 90] Luan S. W. , Gligor V. D., "A Fault-Tolerant Protocol for Atomic Broadcast"; *IEEE Transactions on Parallel and Distributed Systems*, vol 1, No. 3, July 1990, pp 271 - 285.

[Melliar-Smith 90] Melliar-Smith P. M. , Moser L. E. , Ágrawala V. , "Broadcast Protocols for Distributed Systems"; *IEEE Transactions on Parallel and Distributed Systems*, vol 1, No. 1, January 1990, pp 17 - 25.

[Nacamura 92] Nacamura Junior L., "Proposição de um modelo de tolerância a faltas baseado em alta replicação de processos" - *Relatório Técnico RT 92-21*, LCMI - UFSC, Brasil, Abril 1992.

[Nakamura 90] Nakamura A., Hasegawa M., Takizawa M , Katsura S., "Selectively Partially Ordering Broadcast Service for Distributed Systems"; *Proc of the 5th Internacional Joint Workshop Computer Communication*, Kyongju, Korea, July 1990, pp 315 - 324.

[Nishio 90] Nishio S. , Li K. F. , Manning E. G., "A Resilient Mutual Exclusion Algorithm for Computer Network"; *IEEE Transactions on Parallel and Distributed Systems*, vol 1, No. 3, July 1990, pp 344 - 355.

[Pehrson 90] Pehrson B., "Protocol Verification for OSI"; *Computer Networks and ISDN Systems 18*, 1989/90, pp 185 - 201.

[Perry 86] Perry K. J. , Toueg S., "Distributed Agreement in the Presence of Processor and Communication Faults"; *IEEE Transaction on Software Engineering*, vol 3, march 1986, pp 477 - 481

[Powell 90] Powell D., "Fault Assumptions and Assumption Coverage - a Contribuion to the Fundamental Concepts of Dependability", *Rapport LAAS Number 90074*, France, February, 1990.

[Peterson 89] Peterson L. L., Buchholz N. C., Schilichting R. D, "Preserving and Using Context Information in Interprocess Communication"; *ACM Transaction on Computer Systems*, vol 7, No. 3, August 1989, pp 217 - 246.

[Saqui 90] Saqui-Sannes P., "The ESTIM User Manual for release 4.0 on Sun3 & Sun4 machines"; november 1990.

[Segall 83] Segall A. , Awerbuch B. , "A Reliable Broadcast Protocol"; *IEEE Transactions on Communications*, vol COM-31, No. 7, July 1983.

[Shivastava 91] Ezhilchelvan P. D. , Shivastava S. K., "A Distributed Systems Architecture Suporting High Availability and Reliability"; *Proc 2th International Working Conference on Dependable Computing for Critical Applications*, Tucson, Arizona, USA, February 1991, pp 36 - 48.

[Srikant 86] Srikant T. K., "Designing Fault Tolerant Algorithms for Distributed Systems Using Communication Primitives"; *Ph. D. Thesis*, Dpto of Comtuter Science, Cornell University, Ithaca, NY, USA, February 1986.

[Takizawa 87] Takizawa M., "Cluster Control Protocol for Highly Reliable Broadcast Communication"; *Proc. IFIP WG 10.3 Working Conference on Distributed Processing, Amsterdan*, The Nethelands, October 1987, pp 431 - 445.

[Veríssimo 89] Veríssimo P., "Comunicação em Grupo Fiável em Sistemas Distribuídos sobre Rede Local"; *Tese de Doutorado*, INESC, Portugal, Dezembro 1989.

APÊNDICE A

ALGORITMO DE REGENERAÇÃO

Algoritmo de regeneração de token, baseado em [Nishio 91]

SE expira T_{reconf} de E_i para recepção de token *ENTÃO*
(* regeneração do token disparada *)

Versão-proposta := min (m /(m = K_{vt} * N + i)[1] , (m > versão-estação$_i$))
INCREMENTA K_{vt}
versão-estação$_i$:= Versão-proposta$_i$

DIFUNDE token-perdido$_i$[2]
(* comunicação síncrona com sinalização de tarefa, ficando a esperar reconhecimento das demais estações *)

ESPERA até ocorrer uma das condições abaixo:

c1: reconhecimento NEGATIVO for recebido
(* token não regenerado *)
SE causa do reconhecimento negativo for último-commit$_i$ < último commit executado por G *ENTÃO*
Recupera commit(s) perdido(s)

SAI da espera de reconhecimento

c2: reconhecimento POSITIVO for recebido de todas estações ϵ G_p *OU* EXPIRA timeout
SE número de reconhecimentos > (N /2 + 1) *ENTÃO*
(* maioria concorda,ninguem discorda *)
REGENERA token comVersão-token = Versão-proposta$_i$
E_c = E_i
DIFUNDE Regeneração-Token[3]
SENÃO
Token não regenerado

---

[1] K_{vt} - número de vezes que a versão-token foi determinada
i - valor da i-ésima estaçao , estação regeneradora, isto é, que detectou perda de token.

[2] A mensagem Token-perdido$_i$ deve conter:
- A Estação Regeneradora E_i que detectou a perda do token e difundiu a mensagem.
- **Versão-proposta$_i$**, isto é, próxima versão do token.
- **Último-commit$_i$**, isto é, o último commit executado por E_i.

[3] Regeneração-Token é uma mensagem contendo a estação Ec e a versão-token.

Após a reconfigurção do token, caso o anel possua mais de L mensagens: Faz-se o Pedido-de-commit e volta tudo ao normal. Caso o token não seja regenerado, uma nova estação deverá ter seu T_{reconf} expirado e tentará reconfigurar o anel.

Procedure de Resposta ao Token-perdido$_i$ executado por E_j

```
SE Ej recebe mensagem Token-perdidoi ENTÃO
   SE versão-propostai ≤ versão-estaçãoi ENTÃO
   (* versão desatualizada *)
      RECONHECE NEGATIVAMENTE Token-perdidoi
   SENÃO (* versão atualizada *)
      SE Ej não possui o token ENTÃO
         Versão-estaçãoj = versão-propostai
         SE último-commiti >= último-commit de Ej ENTÃO
         (* Estação regeneradora fez todos commits *)
            RECONHECE POSITIVAMENTE token- perdidoi
         SENÃO (* Estação regeneradora não fez todos commits *)
            RECONHECE NEGATIMENTE  token-perdidoi
      SENÃO (* token não foi perdido, Ej possui o token *)
         Versão-estaçãoj = Versão-propostai
         Versão-token = Versão-propostai
         RECONHECE NEGATIMENTE Token-perdidoi.
```

APÊNDICE B

ESPECIFICAÇÃO DO PROTOCOLO PROPOSTO EM PSEUDO-CODIGO

Este apêndice contém um pseudo-código da especificação Estelle do protocolo de difusão atômica proposto. A apresentação de uma pseudo-linguagem, ao invés da especificação Estelle tem por objetivo facilitar o entendimento do protocolo proposto.

```
SPECIFICATION Camada-difusão-confiável

   MODULE Mod-Envio ACTIVITY (estação, coordenadora,Gp )
   END;

   MODULE Mod-Recepção ACTIVITY (estação, coordenadora,Gp )
   END;

   MODULE Mod-Gestão ACTIVITY (estação, coordenadora,Gp )
   END;

   MODULE Mod-Interface-rede ACTIVITY (estação, coordenadora,Gp )
   END;

   BODY Corpo-Envio FOR Mod-Envio;

      STATE recmsg;
      INITIALIZE
         TO recmsg
            BEGIN
               Setar Ki  (* variável do número de mensagens enviadas e não
               engajadas *)
               Setar Fila-das-mensagens
            END;

      TRANS
         FROM recmsg TO recmsg
         WHEN Eaplic.MSG
         PROVIDED (Ki ≤ K)
            BEGIN
               ADICIONA a MSG sua sequência e o nome da estação
               DIFUNDE mensagem MSGDIFUNDIDA
               COLOCA mensagem na Fila-das-mensagens
            END;

      TRANS
         FROM recmsg TO recmsg
         WHEN EG.AVISO-COMMIT
            BEGIN
               RETIRA da Fila-das-mensagens as mensagens engajadas
               DECREMENTA Ki do número de mensagens engajadas
            END;
```

```
   TRANS
      WHEN Pacess.PEDIDO-RETRANSMISSÃO
      PROVIDED (existe a mensagem na Fila-das-mensagens)
         BEGIN
            ENVIA mensagem MSGDIFUNDIDA a estação requisitante
         END;
END;  (* Body corpo-Envio *)


BODY corpo-Recepção FOR Mod-Recepção;

   STATE recepmsg,recupmsg;

   INITIALIZE
      TO recepmsg
         BEGIN
            SETAR Fi para cada estação de Gp
            ZERAR variáveis de mensagens perdidas
            CRIAR variável Nmsg    (* número de mensagens nas Fi *)
         END;

   TRANS
      WHEN DR.MSGDIFUNDIDA
      PROVIDED (emissora pertence a Gp ) AND
               (sequência é a esperada para o emissor)
         BEGIN
            COLOCA mensagem na Fi do emissor
            INCREMENTA sequência esperada para o emissor
            INCREMENTA Nmsg
         END;

   TRANS
      FROM recepmsg TO recepmsg
      WHEN DR.MSGDIFUNDIDA
      PROVIDED (emissora pertence a Gp ) AND
               ( Sk < sequência esperada para o emissor)
         BEGIN
         END;

   TRANS
      TO recupmsg
      WHEN DR.MSGDIFUNDIDA
      PROVIDED (emissora pertence a Gp ) AND
               ( Sk > sequência esperada para o emissor) AND
```

```
            (diferença entre S_k e sequência esperada < K)
      BEGIN
         ARMAZENA mensagem em seu lugar na F_i
         COLOCA emissor no GRUPO falta-msg
         ENVIA Pedido-de-retransmissão da mensagem perdida ao emissor
         INCREMENTA N_m do número de mensagens perdidas mais um
      END;

TRANS
   FROM recupmsg TO recupmsg
   WHEN DR.MSGDIFUNDIDA
   PROVIDED (emissor pertence ao grupo Falta-msg) AND
            (S_k =  sequência esperada do emissor)
      BEGIN
         ARMAZENA mensagem em F_i
         SE (ainda existem mensagens perdidas) ENTÃO
            ENVIA Pedido-de-retransmissão da mensagem perdida ao
                  emissor
         SENÃO
            RETIRA emissor do grupo Falta-msg
      END;

TRANS
   FROM recupmsg TO recupmsg
   PROVIDED (exite estação no grupo Falta-msg)
   DELAY (timeout)
      BEGIN
         ENVIA novamente Pedido-de-retransmissão da mensagem perdida à
         estação emissora
      END;

TRANS
   FROM recupmsg TO recepmsg
   PROVIDED (não exite estação no grupo Falta-msg)
      BEGIN
      END;

TRANS
   FROM recepmsg TO recepmsg
   PROVIDED (completou L mensagens nas F_i) AND
            (estação é a coordenadora)
      BEGIN
         CALCULAR a última mensagem existente nas F_i de cada estação
                  engajante (P_EGj(j))
```

```
            ENVIAR AV-L ao modulo Estelle Coordenação (* AV-L contém o
                                    P_EGi(j) e a indicação da nova coordendora *)
         END;

TRANS
   FROM recepmsg TO recepmsg
   WHEN RG3.PED-L
      (* Estourou timeout e o módulo Estelle Coordenação pediu o
      P_EGi(j) e a nova coordenadora para uma próxima Execução de
      Protocolo *)
      BEGIN
         CALCULAR a última mensagem existente nas F_i de cada estação
                  engajante (P_EGi(j))
         ENVIAR AV-L ao modulo Estelle Coordenação (* AV-L contém o
                                    P_EGi(j) e a indicação da nova coordendora *)
      END;

TRANS
   FROM recepmsg TO recepmsg
   WHEN RG1.PED-VERIFICAÇÃO
      BEGIN
         VERIFICAR   em cada estação engajante E_EGi a existência de TODAS
                     mensagens indicadas no Ped-verificação
         SETAR rec = SIM
         SE existe E_i | G_L, tal que P_Fi for diferente P_EGi(j) ENTÃO
            CASO diferença SEJA
               < 0   (* PERDA DE MENSAGEM *)
                  ENVIAR Pedido-de-retransmissão ao emissor
                        rec = talvez
               > 0   (* REPETIÇÃO DO PONTEIRO *)
                  SE já existiam as mensagens na última vez que a
                     estação pertenceu a G_L ENTÃO
                     rec = NÃO
                  SENÃO
                     rec = SIM
            ENVIA rec ao modulo Estelle Engajamento
      END;

TRANS
   WHEN RG1.ENGAJAMENTO
   PROVIDED (commit é o executado atualmente no anel)
      BEGIN
         ENVIA FILA-ENG com as mensagens das F_i na ordem indicada pela
         mensagem Engajamento para a APLICAÇÃO
```

```
            RETIRA mensagens engajadas das Fi
            DIMINUI do Nm o número de mensagens engajadas
            SE estação pertence ao Grupo-R ENTÃO
               ENVIA FILA-ENG ao modulo Estelle Coordenação
            RETIRA Gf de Gp
         END;

   TRANS
      WHEN RG1.ENGAJAMENTO
      PROVIDED (commit é um commit perdido)
         BEGIN
            ENQUANTO exite estação engajante FAÇA
               SE falta mensagem na Fi ENTÃO
            ENVIA PEDIDO-RETRANSMISSÃO a uma estação do Grupo-R
            ENVIA FILA-ENG com as mensagens das Fi na ordem indicada
                  pela mensagem ENGAJAMENTO para a APLICAÇÃO
            RETIRA mensagens engajadas das Fi
            RETIRA Gf de Gp
            RECONHECE ENGAJAMENTO ao modulo Estelle Coordenação
         END;

   TRANS
      FROM recepmsg TO recepmsg
      WHEN RG3.AV-RETR
         BEGIN
            COMPARA Fi com AV-RETR
            RECUPERA mensagens perdidas
            ENVIA AV-L ao módulo Estelle Coordenação
         END;

   TRANS
      WHEN RG2.PED-MSG
         BEGIN
            ENVIA FILA-REG ao módulo Estelle Coordenação
         END;

END;


BODY corpo-Gestão FOR Mod-Gestão;

   MODULE Mod-Coordenação ACTIVITY;
   END;
```

```
MODULE Mod-Engajamento ACTIVITY;
END;

BODY corpo-Coordenação FOR Mod-Coordenação;

   STATE wait, espera-fila-L, ped-com, ped-com1, abort, commiting;

   INITIALIZE
      TO wait
         BEGIN
            SETAR grupos
         END;

   TRANS
      FROM wait TO ped-com
      WHEN Crec.AV-L
      PROVIDED (estação = coordenadora) AND
               (( G_A = {}) OU (não (G_A * G_p ≤ G_r )))
         BEGIN
            DIFUNDE Pedido-de-commit_j as demais estações
         END;

   TRANS
      FROM wait TO wait
      DELAY (timeout-L, timeout-L)
      PROVIDED (estação = coordenadora)
         BEGIN
            ENVIA PED-L ao modulo Estelle Recepção
         END;

   TRANS
      FROM wait TO ped-com
      WHEN Crec.AV-L
      DELAY (timeout-R, timeout-R)
      PROVIDED (estação = coordenadora) AND
               (existe estações pertencentes a G_r(j - M))
         BEGIN
            DIFUNDE Pedido-de-commit_j as demais estações
         END;

   TRANS
      FROM ped-com TO ped-com1
         BEGIN
```

```
            ARMAZENA em G_obs as estações comum ao G_A das últimas M-1
                    execuções de protocolo
         END;

TRANS
   FROM ped-com1 TO ped-com1
   WHEN DCacess.ACK
   PROVIDED (estação = coordenadora) AND  (sender_ACK ∈ G_p )   AND
            (commit_ACK = commit_j)     AND   (sender_ACK ∉ G_EG(j))
      BEGIN
         COLOCA sender_ACK  em G_EG(j)
         SE número de estações em G_EG(j) < R ENTÃO
            COLOCA sender_ACK no Grupo-R
      END;

TRANS
   FROM ped-com1 TO abort
   WHEN DCacess.NACK-C
   PROVIDED (commit_NACK = commit_j)
      BEGIN
         DIFUNDE ABORT as demais estações
         ENVIA AV-RETR ao modulo Estelle Recepção
      END;

TRANS
   FROM ped-com1 TO commiting
   PROVIDED ( estação = coordenadora)
   DELAY (timeout-ACK, timeout-ACK)
      BEGIN
         CALCULA G_f = ((G_p - G_EG) * G_obs)
      END;

TRANS
   FROM ped-com1 TO commiting
   PROVIDED (G_EG(j) = G_p )
      BEGIN
      END;

TRANS
   FROM commiting TO espera-fila-L
   PROVIDED (G_EG(j) ≥ N/2) AND
            (nova coordenadora proposta pertence ao G_EG(j))
      BEGIN
         CALCULA G_A(j)
```

```
            DIFUNDE Commit para as demais estações
            ARMAZENA mensagem Commit na Lista-M-msg-commit
            ATUALIZA coordenadora
         END;

TRANS
   FROM commiting TO espera-fila-L
   PROVIDED (G_EG(j) ≥ N/2) AND
            (nova coordenadora proposta não pertence ao G_EG(j))
      BEGIN
         CALCULA  nova coordenadora tal que seja a última estação-L
                  de L_me(j) pertencente ao G_EG(j)
         CALCULA  nova L_me(j)
         CALCULA  G_A(j)
         DIFUNDE  Commit para as demais estações
         ARMAZENA mensagem Commit na Lista-M-msg-commit
         ATUALIZA coordenadora
      END;

TRANS
   FROM commiting TO abort
   PROVIDED (G_EG ≤ N/2)
      BEGIN
         DIFUNDE Abort às demais estações
      END;

TRANS
   FROM abort TO wait
   DELAY (timeout,timeout)
      BEGIN
         ENVIA PED-L ao modulo Estelle Recepção
      END;

TRANS
   WHEN DCacess.ACK
   PROVIDED (sender_ACK ∈ G_p ) AND (commit_ACK < commit_j)
      BEGIN
         SE (estação = coordenadora) ENTÃO
            SE (sender_ACK ∈ G_r(j)) ENTÃO
               RETIRA sender_ACK de G_r(j)
            SE (envia-prox-msg) ENTÃO
               ENVIA próximo COMMIT a estação sender_ACK
            SE (sender_ACK ∈ G_A do commit_ACK) ENTÃO
               RETIRA sender_ACK de G_A(commit_ACK)
```

```
            SE (sender_ACK ∈ G_obs) ENTÃO
               RETIRA sender_ACK de G_obs
      END;

TRANS
   WHEN PRacess.PED-COMMIT-PERD
   PROVIDED (estação = coordenadora)
      BEGIN
         ENVIA COMMIT requerido
            COLOCA sender Ped-commit-perd no G_r(commit-perdido)
      END;

TRANS
   WHEN PRacess.PED-COMMIT-PERD
   PROVIDED (estação não é coordenadora)
      BEGIN
         ENVIA Ped-commit-perd a atual coordenadora
      END;

TRANS
   FROM wait TO espera-fila-L
   WHEN RE.COMM
   PROVIDED ( estação pertence ao Grupo-R)
      BEGIN
         COLOCA COMM na Lista-M-msg-commit
         ATUALIZA coordenadora
         RETIRA as estações pertencentes a G_f de G_p
      END;

TRANS
   FROM espera-fila-L TO wait
   WHEN Rrec.FILA-REG
   PROVIDED (N-commit = commit_j)
      BEGIN
         COLOCA mensagens na Lista-Regeneração
      END;

TRANS
   FROM espera-fila-L TO espera-fila-L
   DELAY (tempo1,tempo1)
      BEGIN
         ENVIA PED-MSG ao módulo Estelle Recepção (* pede FILA-REG*)
      END;
```

```
   TRANS
      FROM wait TO wait
      WHEN RE.COMM
      PROVIDED ( estação não pertence ao Grupo-R)
         BEGIN
            COLOCA COMM na Lista-M-msg-commit
            ATUALIZA coordenadora
            RETIRA as estações pertencentes a Gf de Gp
         END;

END;


BODY corpo-Engajamento FOR Mod-Engajamento;


   STATE wait, commiting, abort, recuperação, espera, reccom, reconf;

   INITIALIZE
      TO wait
         BEGIN
            SETAR N-commit para um (* número do próximo commit a ser
                                      executado pela estação *)
         END;

   TRANS
      FROM wait TO wait
      WHEN DEacess.PEDIDO-DE-COMMITj
      PROVIDED (versão-token = versão-estação)  AND
               (é o commit esperado atualmente) AND
               (estação não é coordenadora)
         BEGIN
            ENVIA Pedido-verificação ao módulo Estelle Recepção
         END;


   TRANS
      FROM wait TO commiting
      WHEN DEacess.PEDIDO-DE-COMMITj
      PROVIDED (versão-token = versão-estação)  AND
               (é o commit esperado atualmente) AND
               (estação é coordenadora)
         BEGIN
         END;
```

```
TRANS
   TO recuperação
   WHEN DEacess.PEDIDO-DE-COMMIT_j
   PROVIDED (versão-token = versão-estação)  AND
            (N-commit < commit_j)  AND
            ((N-commit - commit_j) < M)
      BEGIN
         ENVIA Pedido-commit-perdido a estação coordenadora
      END;

TRANS
   FROM wait TO wait
   WHEN DEacess.PEDIDO-DE-COMMIT_j
   PROVIDED (versão-token = versão-estação)  AND
            (N-commit < commit_j) AND
            ((N-commit - commit_j) > M)
      BEGIN
         PEDIR reinserção ao G_p à coordenadora
      END;

TRANS
   FROM wait TO commiting
   WHEN Erec.rec
   PROVIDED (rec = SIM)
      BEGIN
         ENVIA ACK para a coordenadora
      END;

TRANS
   FROM wait TO wait
   WHEN Erec.rec
   PROVIDED (rec = TALVEZ)
      BEGIN
      END;

TRANS
   FROM wait TO abort
   WHEN Erec.rec
   PROVIDED (rec = NÃO)
      BEGIN
         ENVIA NACK-C para a coordenadora
      END;

TRANS
```

```
   FROM commiting TO wait
   WHEN DEacess.COMMIT
   PROVIDED (versão-token = versão-estação) AND
            (commit_j = N_commit)
      BEGIN
         INCREMENTA N-commit
         ENVIA ENGAJAMENTO ao modulo Estelle Recepção
         SE (estação é coordenadora) ENTÃO
            ENVIA COMM ao modulo Estelle Coordenação
          SE (estação é engajante) ENTÃO
            ENVIA AVISO-COMMIT ao modulo Estelle Envio
      END;

TRANS
   FROM commiting TO wait
   WHEN DEacess.ABORT
      BEGIN
      END;

TRANS
   FROM abort TO wait
   WHEN DEacess.ABORT
      BEGIN
      END;

TRANS
   FROM recuperação TO reccom
   WHEN DEacess.COMMIT
   PROVIDED
      BEGIN
         ENVIA ENGAJAMENTO ao modulo Estelle Recepção
         ENVIA COMM ao modulo Estelle Coordenação
         SE (estação é engajante) ENTÃO
            ENVIA AVISO-COMMIT ao modulo Estelle Envio
      END;

TRANS
   FROM reccom TO recuperação
   WHEN Erec.rec
      BEGIN
         SE (N_commit < commit_j) ENTÃO
            ENVIA ACK (envia-prox-msg = TRUE)  a Coordenadora
         SENÃO
            ENVIA ACK (envia-prox-msg = FALSE) a Coordenadora
```

```
      END;

TRANS
   FROM recuperação TO recuperação
   DELAY (timeout-rec, timeout-rec)
      BEGIN
         REENVIA Ped-commit-perdido a coordenadora
      END;

TRANS
   FROM recuperação TO wait
   PROVIDED (N_commit > commit_j)
      BEGIN
      END;

TRANS
   FROM wait TO espera
   WHEN DEacess.COMMIT
   PROVIDED (N_commit = commit_j) AND (versão-token = versão-estação)
      BEGIN
         ENVIA ENGAJAMENTO ao modulo Estelle Recepção
         SE (estação não é coordenadora) ENTÃO
            ENVIA COMM ao modulo Estelle Coordenação
         SE (estação é engajante) ENTÃO
            ENVIA Aviso-commit ao modulo Estelle Envio
      END;

TRANS
   FROM espera TO wait
   WHEN Erec.rec
      BEGIN
         INCREMENTA N-commit
      END;

TRANS
   FROM commiting TO commiting
   DELAY (timeout-ack2, timeout-ack2)
      BEGIN
         ENVIA PED-COMMIT-PERD à estação coordenadora
         envpedcompperd = TRUE
      END;

TRANS
   WHEN DEacess.COMMIT
```

```
        PROVIDED (N_commit > commit_j)
           BEGIN
           END;

     TRANS
        FROM commiting TO reconf
        DELAY (timeout-commit, timeout-commit)
        PROVIDED (envpedcompperd)
           BEGIN
              ENTRA EM RECONFIGURAÇÃO DO SISTEMA
           END;

  END;


  MODVAR
     Engajamento :  Mod-Engajamento;
     Coordenação :  Mod-Coordenação;


  INITIALIZE
     BEGIN
        INIT Engajamento WITH Corpo-Engajamento (estação,coordenadora,G_p);
        INIT Coordenação WITH Corpo-Coordenação (estação,coordenadora,G_p);
     END;

END;


BODY corpo-Interface-rede FOR Mod-Interface-rede;

  INITIALIZE
     BEGIN
     END;


  TRANS
     BEGIN
        FORMATAR mensagens
        ENVIAR ao devido modulo Estelle
     END;

END;
```

```
MODVAR

    Envio           :  Mod-Envio;
    Recepção        :  Mod-Recepção;
    Gestão          :  Mod-Gestão;
    Interface-rede  :  Mod-Interface-rede;


INITIALIZE

    BEGIN
       INIT  Recepção WITH Corpo-Recepção (estação, coordenadora,Gp );
       INIT  Gestão WITH Corpo-Gestão (estação, coordenadora,Gp );
       INIT  Envio WITH Corpo-Envio (estação, coordenadora,Gp );
       INIT  Interface-rede WITH Corpo-Interface-rede (estação,
             coordenadora,Gp);
    END;
END.
```

APÊNDICE C

ESPECIFICAÇÃO DO AMBIENTE PARA VALIDAÇÃO EM PSEUDO–CODIGO

Este apêndice contém o pseudo-código da especificação Estelle* do ambiente proposto para a simulação. A apresentação de uma pseudo-linguagem, tem o mesmo objetivo e características que as do apêndice B.

```
SPECIFICATION  Ambiente-para-simulação SYSTEMACTIVITY;

   MODULE Mod-Camada-difusão-confiável ACTIVITY (estação coordenadora, Gp);
   END;

   MODULE Mod-Aplicação ACTIVITY;
   END;

   MODULE Mod-Interface-rede ACTIVITY (Gp);
   END;


   BODY Corpo-Aplicação FOR Mod-Aplicação;

      INITIALIZE
      TO começa
         BEGIN
            CRIAR uma lista de mensagens a enviar
         END;

         TRANS
            WHEN apvt.REQ
               BEGIN
                  ENVIA MSG ao módulo Estelle Envio
               END;

         TRANS
            WHEN ap1.FILA-ENG
         BEGIN
            ENGAJAR  mensagens contidas em FILA-ENG
         END;

   END;


   BODY Corpo-camada-difusão-confiável FOR Mod-Camada-difusão-
         confiável

      (* Especificação no apêndice A *)

   END;
```

```
BODY Corpo-Interface-rede FOR Mod-Interface-rede

   MODULE Mod-Multiponto ACTIVITY (Gp);
   END;

   MODULE Mod-Ponto-a-ponto ACTIVITY (Gp);
   END;

   MODULE Mod-Perturbação ACTIVITY (Gp);
   END;


   BODY Corpo-Multiponto FOR Mod-Multiponto;

      STATE normal, comperdamsg;

      INITIALIZE
         TO normal
            BEGIN
            END;

      TRANS
         FROM normal TO normal
         WHEN drec[i].MSG1 (* onde i é qualquer estação de Gp *)
            BEGIN
               SE MSG1 = COMMIT ENTÃO ATUALIZA Gp
               DIFUNDE MSG1 a todas estações pertencentes a Gp
            END;

      TRANS
         FROM normal TO comperdamsg
         WHEN dperd.PDMSG
            BEGIN
               tipoperdido = PDMSG;
               site = PDMSG.site;
            END;

      TRANS
         FROM comperdamsg TO normal
         WHEN drec[i].MSG1 (* onde i é qualquer estação de Gp *)
         PROVIDED (mensagem recebida é do tipoperdido)
            BEGIN
               SE MSG1 = COMMIT ENTÃO ATUALIZA Gp
```

```
            DIFUNDE MSG1 a todas estações pertencentes a Gp exceto Ei
                        pertencente a Grupo-site
            ENVIA REALIZ ao módulo Estelle Perturbação
         END;


   TRANS
      FROM comperdamsg TO comperdamsg
      WHEN drec[i].MSG1 (* onde i é qualquer estação de Gp *)
      PROVIDED ( mensagem recebida não é do tipoperdido )
         BEGIN
            SE MSG1 = COMMIT ENTÃO ATUALIZA Gp
            DIFUNDE MSG1 a todas estações pertencentes a Gp
         END;

END;


BODY Corpo-Ponto-a-ponto FOR Mod-ponto-a-ponto;

   INITIALIZE
      BEGIN
      END;

   TRANS
      WHEN prec[i].MSG3 (* onde i é qualquer estação de G *)
         BEGIN
            ENVIA MSG3 à estação recomendada;
         END;

END;


BODY Corpo-Perturbação FOR Mod-Perturbação;

   STATE wait, go;

   INITIALIZE
      TO go
         BEGIN
         END;

   TRANS
      FROM go TO wait
```

```
      BEGIN
         SETA Grupo-site com as estações desejadas;
         ENVIA PDMSG(perde-mensagem, Grupo-site) ao módulo Estelle
                  Multiponto;
      END;

TRANS
   FROM go TO wait
      BEGIN
         SETA Grupo-site com as estações desejadas;
         ENVIA PDMSG(perde-commit, Grupo-site) ao módulo Estelle
         Multiponto;
      END;

TRANS
   FROM go TO wait
      BEGIN
         SETA Grupo-site com as estações desejadas;
         ENVIA PDMSG(perde-pedido-de-commit, Grupo-site) ao módulo
                  Estelle Multiponto;
      END;

TRANS
   FROM go TO wait
      BEGIN
         SETA Grupo-site com as estações desejadas;
         ENVIA PDMSG(perde-pedido-de-commit-e-commit, Grupo-site)
         ao módulo Estelle Multiponto;
      END;

TRANS
   FROM go TO wait
      BEGIN
         SETA Grupo-site com as estações desejadas;
         SETA Npm com o valor de mensagens que se deseja perder;
         ENVIA PDMSG(perde-Npm-mensagens, Grupo-site) ao módulo
                  Estelle Multiponto;
      END;

TRANS
   FROM wait TO go
   WHEN Pertdif.REALIZ
      BEGIN
      END;
```

```
    END;

    MODVAR

       Perturbação    :  Mod-Perturbação;
       Multiponto     :  Mod-Multiponto;
       Ponto-a-Ponto  :  Mod-Ponto-a-Ponto;

    INITIALIZE
       BEGIN
          INIT Perturbação WITH Corpo-Perturbação;
          INIT Multiponto WITH Corpo-Multiponto;
          INIT Ponto-a-Ponto WITH Corpo-Ponto-a-Ponto;
       END;

END;


MODVAR

   Aplicação1,
   Aplicação2,
   ...,
   AplicaçãoN                        :  Mod-Aplicação;

   Camada-difusão-confiável1,
   Camada-difusão-confiável2,
   ...,
   Camada-difusão-confiávelN         :  Mod-Camada-difusão-confiável;

   Sistema-de-Comunicação            :  Mod-Sistema-de-Comunicação;


INITIALIZE
   BEGIN
      INIT Aplicaçãoi WITH Corpo-Aplicação;
      INIT Camada-difusão-confiáveli WITH Corpo-Camada-difusão-
           confiável;
      INIT Sistema-de-Comunicação WITH Corpo-Sistema-de-Comunicação;
   END;

END.
```